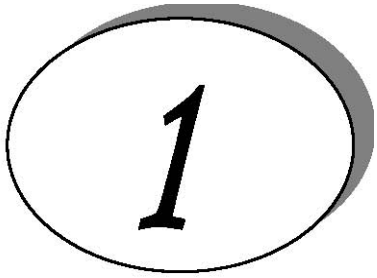
Technical Aptitude Questions

Face Interviews Confidently!

Table of Contents

Data Structures Aptitude



1. What is data structure?

A data structure is a way of organizing data that considers not only the items stored, but also their relationship to each other. Advance knowledge about the relationship between data items allows designing of efficient algorithms for the manipulation of data.

2. List out the areas in which data structures are applied extensively?

Compiler Design,
Operating System,
Database Management System,
Statistical analysis package,
Numerical Analysis,
Graphics,
Artificial Intelligence,
Simulation

3. What are the major data structures used in the following areas : RDBMS, Network data model & Hierarchical data model.

RDBMS – Array (i.e. Array of structures)
Network data model – Graph
Hierarchical data model – Trees

4. If you are using C language to implement the heterogeneous linked list, what pointer type will you use?

The heterogeneous linked list contains different data types in its nodes and we need a link, pointer to connect them. It is not possible to use ordinary pointers for this. So we go for void pointer. Void pointer is capable of storing pointer to any type as it is a generic pointer type.

5. Minimum number of queues needed to implement the priority queue?

Two. One queue is used for actual storing of data and another for storing priorities.

6. What is the data structures used to perform recursion?

Stack. Because of its LIFO (Last In First Out) property it remembers its 'caller' so knows whom to return when the function has to return. Recursion makes use of system stack for storing the return addresses of the function calls.

Every recursive function has its equivalent iterative (non-recursive) function. Even when such equivalent iterative procedures are written, explicit stack is to be used.

7. What are the notations used in Evaluation of Arithmetic Expressions using prefix and postfix forms?

Polish and Reverse Polish notations.

*8. Convert the expression ((A + B) * C – (D – E) ^ (F + G)) to equivalent Prefix and Postfix notations.*

Prefix Notation:

^ - * +ABC - DE + FG Postfix Notation:

AB + C * DE - - FG + ^

9. Sorting is not possible by using which of the following methods?

(a) Insertion
(b) Selection
(c) Exchange
(d) Deletion

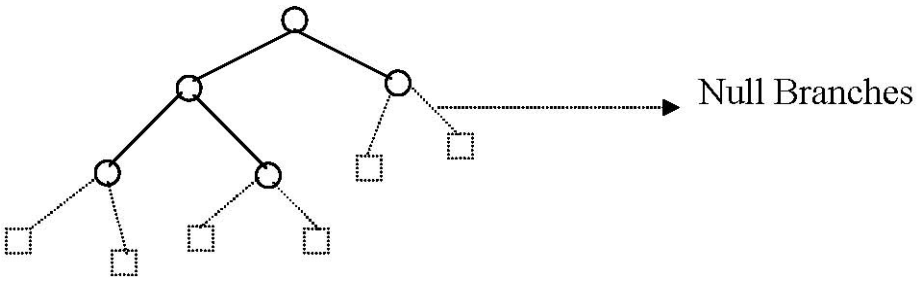
(d) Deletion.

Using insertion we can perform insertion sort, using selection we can perform selection sort, using exchange we can perform the bubble sort (and other similar sorting methods). But no sorting method can be done just using deletion.

10. A binary tree with 20 nodes has null branches?

21

Let us take a tree with 5 nodes (n=5)

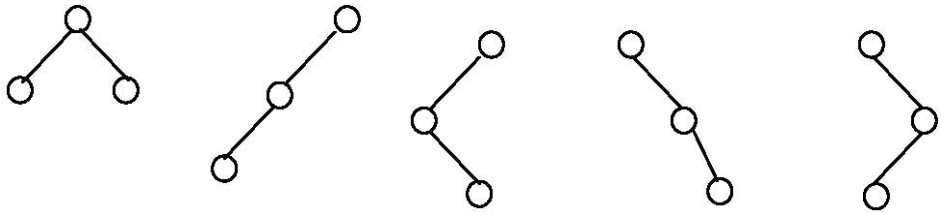


It will have only 6 (ie,5+1) null branches. In general,
*A binary tree with **n** nodes has exactly **n+1** null nodes.*

11. What are the methods available in storing sequential files ?
Straight merging,
Natural merging,
Polyphase sort,
Distribution of Initial runs.

12. How many different trees are possible with 10 nodes ?

1014
For example, consider a tree with 3 nodes(n=3), it will have the maximum combination of 5 different (ie, $2^3 - 3 = 5$) trees.



i ii iii iv v

In general:
*If there are **n** nodes, there exist $2^n - n$ different trees.*

13. List out few of the Application of tree data-structure?
The manipulation of Arithmetic expression,
Symbol Table construction,

Syntax analysis.

14. List out few of the applications that make use of Multilinked Structures?

Sparse matrix,
Index generation.

15. In tree construction which is the suitable efficient data structure?

(a) Array (b) Linked list (c) Stack (d) Queue (e) none

(b) Linked list

16. What is the type of the algorithm used in solving the 8 Queens problem?

Backtracking

17. In an AVL tree, at what condition the balancing is to be done?

If the 'pivotal value' (or the 'Height factor') is greater than 1 or less than –1.

18. What is the bucket size, when the overlapping and collision occur at same time?

One. If there is only one entry possible in the bucket, when the collision occurs, there is no way to accommodate the colliding value. This results in the overlapping of values.

19. Traverse the given tree using Inorder, Preorder and Postorder traversals.

Inorder : D H B E A F C I G J
Preorder: A B D H E C F G I J

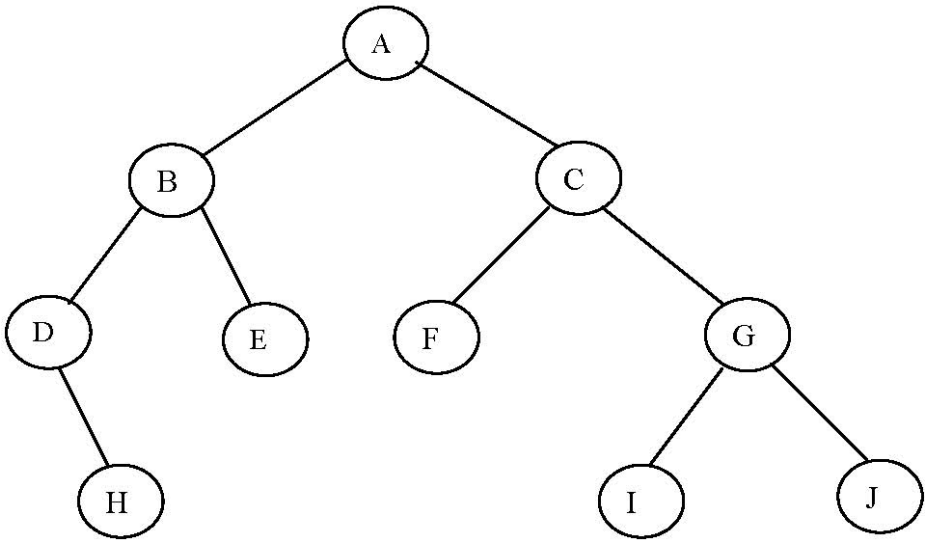
Postorder: H D E B F I J G C A

20. There are 8, 15, 13, 14 nodes were there in 4 different trees. Which of them could have formed a full binary tree?

15.

In general:

Given tree:



There are $2^n - 1$ *nodes in a full binary tree.*

By the method of elimination:

Full binary trees contain *odd* number of nodes. So there cannot be full binary trees with 8 or 14 nodes, so rejected. With 13 nodes you can form a *complete* binary tree but not a full binary tree. So the correct answer is 15.

Note:

Full and Complete binary trees are different. *All full binary trees are complete binary trees but not vice versa.*

21. In the given binary tree, using array you can store the node 4 at which location?

At location 6

1	2	3	-	-	4	-	-	5

Root LC1 RC1 LC2 RC2 LC3 RC3 LC4 RC4

where LCn means Left Child of node n and RCn means Right Child of node n

22. Sort the given values using Quick Sort?

65 70 75 80 85 60 55 50 45

Sorting takes place from the pivot value, which is the first value of the given elements, this is marked bold. The values at the left pointer and right pointer are indicated using ^{L} and ^{R} respectively.

65 70^{L} 75 80 85 60 55 50 45^{R}

Since pivot is not yet changed the same process is continued after interchanging the values at ^{L} and ^{R} positions

65 45 75^{L} 80 85 60 55 50^{R} 70

65 45 50 80^{L} 85 60 55^{R} 75 70

65 45 50 55 85^{L} 60^{R} 80 75 70

65 45 50 55 60^{R} 85^{L} 80 75 70

When the L and R pointers cross each other the pivot value is interchanged with the value at right pointer. If the pivot is changed it means that the pivot has occupied its original position in the sorted order (shown in bold italics) and hence two different arrays are formed, one from start of the original array to the pivot position-1 and the other from pivot position+1 to end.

60^{L} 45 50 55^{R} ***65*** **85**^{L} 80 75 70^{R}

55^{L} 45 50^{R} ***60*** ***65*** **70**^{R} 80^{L} 75 ***85***

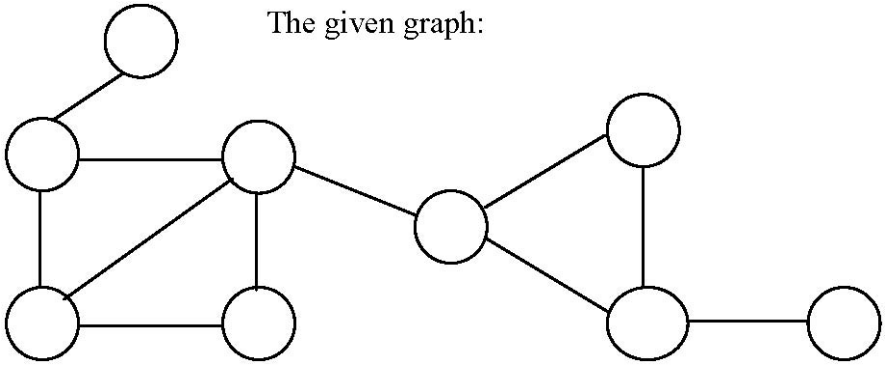
50^{L} 45^{R} ***55*** ***60*** ***65*** ***70*** 80^{L} 75^{R} ***85***

In the next pass we get the sorted form of the array.

45 50 55 60 65 70 75 80 85

23. For the given graph, draw the DFS and BFS?

The given graph:



BFS: A X G H P E M Y J

DFS: A X H P E Y M J G

24. Classify the Hashing Functions based on the various methods by which the key value is found.

Direct method,

Subtraction method,

Modulo-Division method,
Digit-Extraction method,
Mid-Square method,
Folding method,
Pseudo-random method.

25. What are the types of Collision Resolution Techniques and the methods used in each of the type?

Open addressing (closed hashing),
The methods used include:
Overflow block,
Closed addressing (open hashing)
The methods used include:
Linked list,
Binary tree…

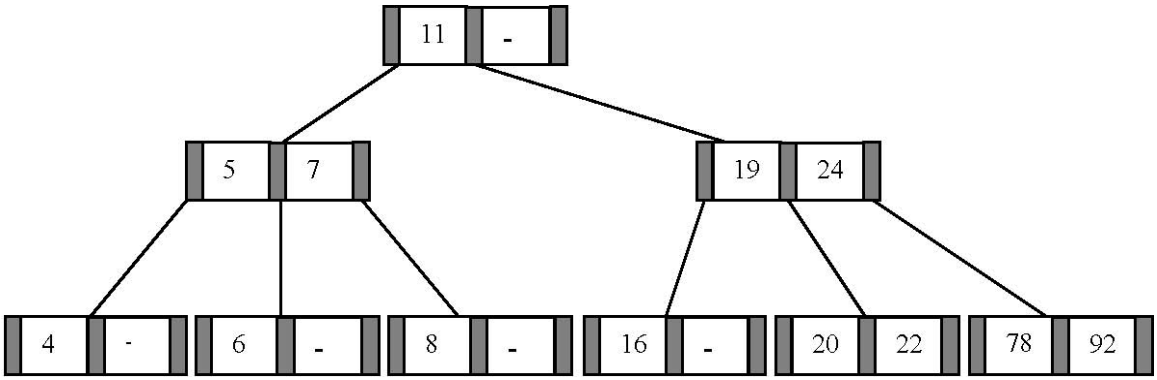
26. In RDBMS, what is the efficient data structure used in the internal storage representation?

B+ tree. Because in B+ tree, all the data is stored only in leaf nodes, that makes searching easier. This corresponds to the records that shall be stored in leaf nodes.

27. Draw the B-tree of order 3 created by inserting the following data arriving in sequence – 92 24 6 7 11 8 22 4 5 16 19 20 78

28. Of the following tree structure, which is, efficient considering space and time complexities?

(a) Incomplete Binary Tree
(b) Complete Binary Tree
(c) Full Binary Tree



(b) Complete Binary Tree.

By the method of elimination:

Full binary tree loses its nature when operations of insertions and deletions are done. For incomplete binary trees, extra storage is required and overhead of NULL node checking takes place. So complete binary tree is the better one since the property of complete binary tree is maintained even after operations like additions and deletions are done on it.

29. What is a spanning Tree?

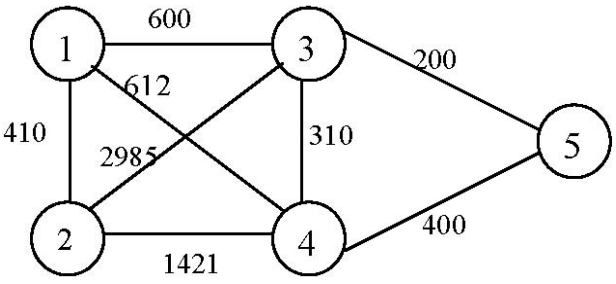
A spanning tree is a tree associated with a network. All the nodes of the graph appear on the tree once. A minimum spanning tree is a spanning tree organized so that the total edge weight between nodes is minimized.

30. Does the minimum spanning tree of a graph give the shortest distance between any 2 specified nodes?

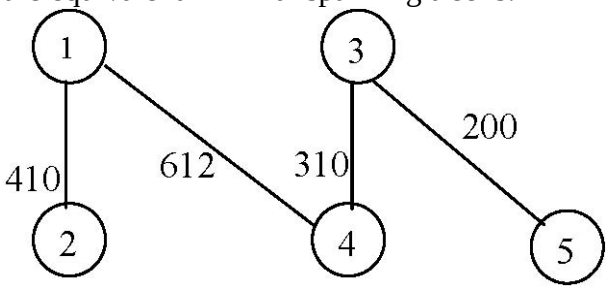
No.

Minimal spanning tree assures that the total weight of the tree is kept at its minimum. But it *doesn't* mean that the distance between any two nodes involved in the minimum-spanning tree is minimum.

31. Convert the given graph with weighted edges to minimal spanning tree.



the equivalent minimal spanning tree is:



32. Which is the simplest file structure?

(a) Sequential
(b) Indexed
(c) Random

(a) Sequential

33. Whether Linked List is linear or Non-linear data structure?

According to Access strategies Linked list is a linear one.
According to Storage Linked List is a Non-linear one.

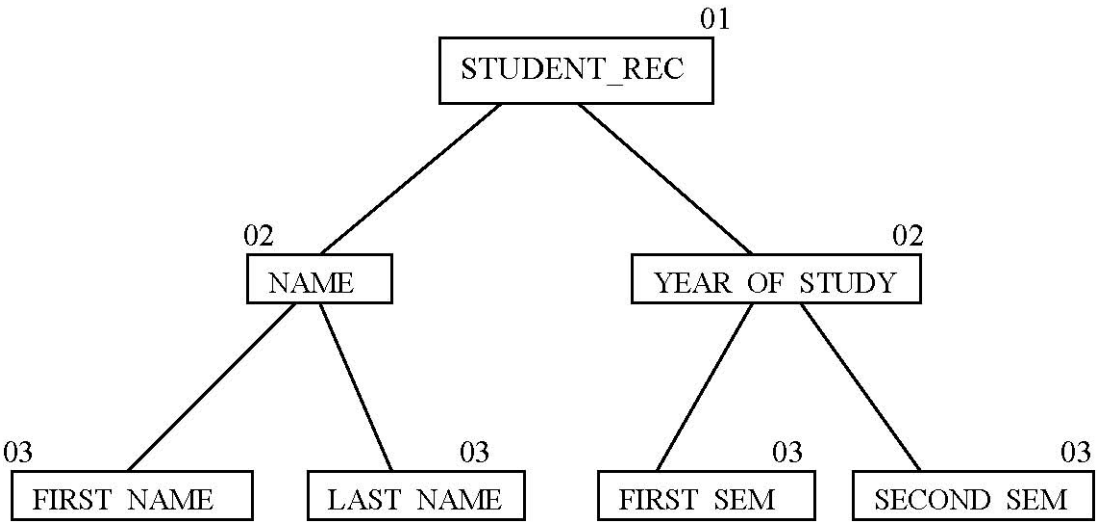
34. Draw a binary Tree for the expression :

*A * B - (C + D) * (P / Q)*

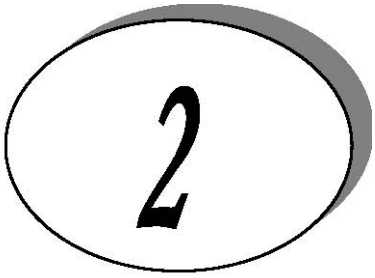
35. For the following COBOL code, draw the Binary tree?

```
01 STUDENT_REC.
      02 NAME.
            03 FIRST_NAME PIC X(10).
      03 LAST_NAME PIC X(10).

 02 YEAR_OF_STUDY.
      03 FIRST_SEM PIC XX.
      03 SECOND_SEM PIC XX.
```



C Aptitude



Note : All the programs are tested under Turbo C/C++ compilers.
It is assumed that,
Programs run under DOS environment,
The underlying machine is an x86 system,
Program is compiled using Turbo C/C++ compiler.

The program output may depend on the information based on this assumptions (for example sizeof(int) == 2 may be assumed).

Predict the output or error(s) for the following:

1. void main()
{

```
int   const * p=5;
printf("%d",++(*p));
```

}

Answer:

Compiler error: Cannot modify a constant value.

Explanation:

p is a pointer to a "constant integer". But we tried to change the value of the "constant integer".

2. main()
{

```
char s[ ]="man";
int i;
for(i=0;s[ i ];i++)
```

```
printf("\n%c%c%c%c",s[ i ],*(s+i),*(i+s),i[s]);
   }
```

Answer:

mmmm

aaaa

nnnn

Explanation:

s[i], *(i+s), *(s+i), i[s] are all different ways of expressing the same idea. Generally array name is the base address for that array. Here **s** is the base address. **i** is the index number/displacement from the base address. So, indirecting it with * is same as s[i]. i[s] may be surprising. But in the case of C it is same as s[i].

3. main()

```
   {

float me = 1.1;
double you = 1.1;
if(me==you)
            printf("I love U");
      else
 printf("I hate U");
   }
```

Answer:

I hate U

Explanation:

For floating point numbers **(**float, double, long double**)** the values cannot be predicted exactly. Depending on the number of bytes, the precession with of the value represented varies. Float takes 4 bytes and long double takes 10 bytes. So float stores 0.9 with less precision than long double.

Rule of Thumb:

Never compare or at-least be cautious when using floating point numbers with relational operators **(**== , >, <, <=, >=,!= **)** .

4. main()

```
{
static int var = 5;
printf("%d ",var--);
if(var)
 main();
}
```

Answer:

5 4 3 2 1

Explanation:

When *static* storage class is given, it is initialized once. The change in the value of a *static* variable is retained even between the function calls. Main is also treated like any other ordinary function, which can be called recursively.

5. main()

```
    {

  int c[ ]={2.8,3.4,4,6.7,5};
  int j,*p=c,*q=c;
  for(j=0;j<5;j++) {
  printf(" %d ",*c);
      ++q;     }
  for(j=0;j<5;j++){
                printf(" %d ",*p);
                ++p;     }
    }
```

Answer:

2 2 2 2 2 2 3 4 6 5

Explanation:

Initially pointer c is assigned to both **p** and **q**. In the first loop, since only **q** is incremented and not c , the value 2 will be printed 5 times. In second loop **p** itself is incremented. So the values 2 3 4 6 5 will be printed.

6. main()

```
    {

 extern int i;
 i=20;
        printf("%d",i);
    }
```

Answer:

Linker Error : Undefined symbol '_i'

Explanation:

extern storage class in the following declaration,

extern int i;

specifies to the compiler that the memory for **i** is allocated in some other program and that address will be given to the current program at the time of linking. But linker finds that no other variable of name **i** is available in any other program with memory space allocated for it. Hence a linker error has occurred .

7. main()

```
    {

 int i=-1,j=-1,k=0,l=2,m;
 m=i++&&j++&&k++||l++;
 printf("%d %d %d %d %d",i,j,k,l,m);
    }
```

Answer:

0 0 1 3 1

Explanation :

Logical operations always give a result of **1 or 0** . And also the logical AND (&&) operator has higher priority over the logical OR (||) operator. So the expression **'i++ && j++ && k++'** is executed first. The result of this expression is 0 (-1 && -1 && 0 = 0). Now the expression is 0 || 2 which evaluates to 1 (because OR operator always gives 1 except for '0 || 0' combination- for which it gives 0). So the value of m is 1. The values of other variables are also incremented by 1.

8. main()

```
{

char *p;
printf("%d %d ",sizeof(*p),sizeof(p));
}
```

Answer:

1 2

Explanation:

The sizeof() operator gives the number of bytes taken by its operand. P is a character pointer, which needs one byte for storing its value (a character). Hence sizeof(*p) gives a value of 1. Since it needs two bytes to store the address of the character pointer sizeof(p) gives 2.

9. main()

```
{

int i=3;
switch(i)
 {
    default:printf("zero");
    case 1: printf("one");
    break;
   case 2:printf("two");
   break;
  case 3: printf("three");
   break;
  }
   }
```

Answer :

three

Explanation :

The default case can be placed anywhere inside the loop. It is executed only when all other cases doesn't match.

10. main()

```
{

printf("%x",-1<<4);
}
```

Answer:

fff0

Explanation :

-1 is internally represented as all 1's. When left shifted four times the least significant 4 bits are filled with 0's.The %x format specifier specifies that the integer value be printed as a hexadecimal value.

11. main()

```
{

char string[]="Hello World";
display(string);
}
void display(char *string)
{
printf("%s",string);
}
```

Answer:

Compiler Error : Type mismatch in redeclaration of function display

Explanation :

In third line, when the function **display** is encountered, the compiler doesn't know anything about the function display. It assumes the arguments and return types to be integers, (which is the default type). When it sees the actual function **display,** the arguments and type contradicts with what it has assumed previously. Hence a compile time error occurs.

12. main()

```
{

int c=- -2;
printf("c=%d",c);
}
```

Answer:

c=2;

Explanation:

Here unary minus (or negation) operator is used twice. Same maths rules applies, ie. minus * minus= plus.

Note:

However you cannot give like --2. Because -- operator can only be applied to variables as a **decrement** operator (eg., i--). 2 is a constant and not a variable.

13. #define int char

```
main()
{

int i=65;
printf("sizeof(i)=%d",sizeof(i));
}
```

Answer:

sizeof(i)=1

Explanation:

Since the #define replaces the string **int** by the macro **char**

14.
```
main()
{
    int i=10;
    i=!i>14;
    printf("i=%d",i);
}
```

Answer:

i=0

Explanation:

In the expression **!i>14** , NOT (!) operator has more precedence than ‘ >’ symbol. **!** is a unary logical operator. !i (!10) is 0 (not of true is false). 0>14 is false (zero).

15.
```
#include<stdio.h>
main()
{
    char s[]={'a','b','c','\n','c','\0'};
    char *p,*str,*str1;
    p=&s[3];
    str=p;
    str1=s;
    printf("%d",++*p + ++*str1-32);
}
```

Answer:

77

Explanation:

p is pointing to character '\n'. str1 is pointing to character 'a' ++*p. "p is pointing to '\n' and that is incremented by one." the ASCII value of '\n' is 10, which is then incremented to 11. The value of ++*p is 11. ++*str1, str1 is pointing to 'a' that is incremented by 1 and it becomes 'b'. ASCII value of 'b' is 98.

Now performing (11 + 98 – 32), we get 77("M");

So we get the output 77 :: "M" (Ascii is 77).

16.
```
#include<stdio.h>
main()
{
    int a[2][2][2] = { {10,2,3,4}, {5,6,7,8}   };
    int *p,*q;
    p=&a[2][2][2];
    *q=***a;
    printf("%d----%d",*p,*q);
}
```

Answer:

SomeGarbageValue---1

Explanation:

p=&a[2][2][2] you declare only two 2D arrays, but you are trying to access the third 2D(which you are not declared) it will print garbage values. *q=***a starting address of a is assigned integer pointer. Now q is pointing to starting address of a. If you print *q, it will print first element of 3D array.

17.
```
#include<stdio.h>
main()
{
    struct xx
    {
        int x=3;
        char name[]="hello";
     };
    struct xx *s;
    printf("%d",s->x);
    printf("%s",s->name);
}
```

Answer:

Compiler Error

Explanation:

You should not initialize variables in declaration

18.
```
#include<stdio.h>
main()
{
    struct xx
    {
        int x;
        struct yy
        {
            char s;
      struct xx *p;
```

```
        };
        struct yy *q;
    };
}
```

Answer:

Compiler Error

Explanation:

The structure yy is nested within structure xx. Hence, the elements are of yy are to be accessed through the instance of structure xx, which needs an instance of yy to be known. If the instance is created after defining the structure the compiler will not know about the instance relative to xx. Hence for nested structure yy you have to declare member.

19.
```
main()
{
    printf("\nab");
    printf("\bsi");
    printf("\rha");
}
```

Answer:

hai

Explanation:

\n - newline
\b - backspace
\r - linefeed

20.
```
main()
{
    int i=5;
    printf("%d%d%d%d%d%d",i++,i--,++i,--i,i);
}
```

Answer:

45545

Explanation:

The arguments in a function call are pushed into the stack from left to right. The evaluation is by popping out from the stack. and the evaluation is from right to left, hence the result.

21.
```
#define square(x) x*x
main()
{
    int i;
    i = 64/square(4);
    printf("%d",i);
}
```

Answer:

64

Explanation:

the macro call square(4) will substituted by 4*4 so the expression becomes i = 64/4*4 . Since / and * has equal priority the expression will be evaluated as (64/4)*4 i.e. 16*4 = 64

22. main()

```
{
    char *p="hai friends",*p1;
    p1=p;
    while(*p!='\0') ++*p++;
    printf("%s    %s",p,p1);
}
```

Answer:

ibj!gsjfoet

Explanation:

++*p++ will be parse in the given order
*p that is value at the location currently pointed by p will be taken
++*p the retrieved value will be incremented
when ; is encountered the location will be incremented that is p++ will be executed

Hence, in the while loop initial value pointed by p is 'h', which is changed to 'i' by executing ++*p and pointer moves to point, 'a' which is similarly changed to 'b' and so on. Similarly blank space is converted to '!'. Thus, we obtain value in p becomes "ibj!gsjfoet" and since p reaches '\0' and p1 points to p thus p1doesnot print anything.

23. #include <stdio.h>

```
#define a 10
main()
{
    #define a 50
    printf("%d",a);
}
```

Answer:

50

Explanation:

The preprocessor directives can be redefined anywhere in the program. So the most recently assigned value will be taken.

24. #define clrscr() 100

```
main()
{
    clrscr();
    printf("%d\n",clrscr());
}
```

Answer:

100

Explanation:

Preprocessor executes as a seperate pass before the execution of the compiler. So textual replacement of clrscr() to 100 occurs.The input program to compiler looks like this :

```
main()
{
      100;
      printf("%d\n",100);
}
```

Note:

100; is an executable statement but with no action. So it doesn't give any problem

25. main()

```
{
    printf("%p",main);
}
```

Answer:

Some address will be printed.

Explanation:

Function names are just addresses (just like array names are addresses).
main() is also a function. So the address of function main will be printed. %p in printf specifies that the argument is an address. They are printed as hexadecimal numbers.

27) main()

```
{
clrscr();
}
clrscr();
```

Answer:

No output/error

Explanation:

The first clrscr() occurs inside a function. So it becomes a function call. In the second clrscr(); is a function declaration (because it is not inside any function).

28) enum colors {BLACK,BLUE,GREEN}

```
 main()
{

 printf("%d..%d..%d",BLACK,BLUE,GREEN);

 return(1);
}
```

Answer:

0..1..2

Explanation:

enum assigns numbers starting from 0, if not explicitly defined.

29) void main()

```
{
 char far *farther,*farthest;

 printf("%d..%d",sizeof(farther),sizeof(farthest));

 }
```

Answer:

4..2

Explanation:

the second pointer is of char type and not a far pointer

30) main()

```
{
 int i=400,j=300;
 printf("%d..%d");
}
```

Answer:

400..300

Explanation:

printf takes the values of the first two assignments of the program. Any number of printf's may be given. All of them take only the first two values. If more number of assignments given in the program,then printf will take garbage values.

31) main()

```
{
 char *p;
 p="Hello";
 printf("%c\n",*&*p);
}
```

Answer:

H

Explanation:

* is a dereference operator & is a reference operator. They can be applied any number of times provided it is meaningful. Here p points to the first character in the string "Hello". *p dereferences it and so its value is H. Again & references it to an address and * dereferences it to the value H.

32) main()

```
{
    int i=1;
```

```
        while (i<=5)
        {
            printf("%d",i);
            if (i>2)
        goto here;
            i++;
        }
}
fun()
{
    here:
      printf("PP");
}
```

Answer:

Compiler error: Undefined label 'here' in function main

Explanation:

Labels have functions scope, in other words The scope of the labels is limited to functions . The label 'here' is available in function fun() Hence it is not visible in function main.

33)

```
main()
{
    static char names[5][20]={"pascal","ada","cobol","fortran","perl"};
     int i;
     char *t;
     t=names[3];
     names[3]=names[4];
     names[4]=t;
     for (i=0;i<=4;i++)
      printf("%s",names[i]);
}
```

Answer:

Compiler error: Lvalue required in function main

Explanation:

Array names are pointer constants. So it cannot be modified.

34)

```
void main()
{
 int i=5;
 printf("%d",i++ + ++i);
}
```

Answer:

Output Cannot be predicted exactly.

Explanation:

Side effects are involved in the evaluation of i

35) void main()
```
{
 int i=5;
 printf("%d",i+++++i);
}
```
Answer:

Compiler Error

Explanation:

The expression i+++++i is parsed as i ++ ++ + i which is an illegal combination of operators.

36) #include<stdio.h>
```
main()
{
int i=1,j=2;
switch(i)
 {
 case 1:   printf("GOOD");
       break;
 case j:   printf("BAD");
       break;
 }
}
```
Answer:

Compiler Error: Constant expression required in function main.

Explanation:

The case statement can have only constant expressions (this implies that we cannot use variable names directly so an error).

Note:

Enumerated types can be used in case statements.

37) main()
```
{
int i;
printf("%d",scanf("%d",&i));   // value 10 is given as input here
}
```
Answer:

1

Explanation:

Scanf returns number of items successfully read and not 1/0. Here 10 is given as input which should have been scanned successfully. So number of items read is 1.

38) #define f(g,g2) g##g2
```
main()
{
int var12=100;
```

```
        printf("%d",f(var,12));
 }
```

Answer:

100

```
39) main()
        {
        int i=0;

        for(;i++;printf("%d",i)) ;
                printf("%d",i);
        }
```

Answer:

1

Explanation:

before entering into the for loop the checking condition is "evaluated". Here it evaluates to 0 (false) and comes out of the loop, and i is incremented (note the semicolon after the for loop).

```
40) #include<stdio.h>
        main()
        {
          char s[]={'a','b','c','\n','c','\0'};
          char *p,*str,*str1;
          p=&s[3];
          str=p;
          str1=s;
          printf("%d",++*p + ++*str1-32);
        }
```

Answer:

M

Explanation:

p is pointing to character '\n'.str1 is pointing to character 'a' ++*p meAnswer:"p is pointing to '\n' and that is incremented by one." the ASCII value of '\n' is 10. then it is incremented to 11. the value of ++*p is 11. ++*str1 meAnswer:"str1 is pointing to 'a' that is incremented by 1 and it becomes 'b'. ASCII value of 'b' is 98. both 11 and 98 is added and result is subtracted from 32.
i.e. (11+98-32)=77("M");

```
41) #include<stdio.h>
        main()
        {
          struct xx
           {
              int x=3;
              char name[]="hello";
```

```
    };
struct xx *s=malloc(sizeof(struct xx));
printf("%d",s->x);
printf("%s",s->name);
}
```

Answer:

Compiler Error

Explanation:

Initialization should not be done for structure members inside the structure declaration

42)
```
#include<stdio.h>
main()
{
struct xx
 {
int x;
struct yy
 {
   char s;
   struct xx *p;
 };
          struct yy *q;
        };
     }
```

Answer:

Compiler Error

Explanation:

in the end of nested structure yy a member have to be declared.

43)
```
main()
{
 extern int i;
 i=20;
 printf("%d",sizeof(i));
}
```

Answer:

Linker error: undefined symbol '_i'.

Explanation:

extern declaration specifies that the variable i is defined somewhere else. The compiler passes the external variable to be resolved by the linker. So compiler doesn't find an error. During linking the linker searches for the definition of i. Since it is not found the linker flags an error.

44)
```
main()
{
```

```
printf("%d", out);
}
int out=100;
```

Answer:

Compiler error: undefined symbol out in function main.

Explanation:

The rule is that a variable is available for use from the point of declaration. Even though a is a global variable, it is not available for main. Hence an error.

45)
```
main()
{
 extern out;
 printf("%d", out);
}
 int out=100;
```

Answer:

100

Explanation:

This is the correct way of writing the previous program.

46)
```
main()
{
 show();
}
void show()
{
 printf("I'm the greatest");
}
```

Answer:

Compier error: Type mismatch in redeclaration of show.

Explanation:

When the compiler sees the function show it doesn't know anything about it. So the default return type (ie, int) is assumed. But when compiler sees the actual definition of show mismatch occurs since it is declared as void. Hence the error.

The solutions are as follows:

1. declare void show() in main() .
2. define show() before main().
3. declare extern void show() before the use of show().

47)
```
main( )
{
   int a[2][3][2] = {{{2,4},{7,8},{3,4}},{{2,2},{2,3},{3,4}}};
   printf("%u %u %u %d \n",a,*a,**a,***a);
printf("%u %u %u %d \n",a+1,*a+1,**a+1,***a+1);
}
```

Answer:

100, 100, 100, 2
114, 104, 102, 3

Explanation:

The given array is a 3-D one. It can also be viewed as a 1-D array.

2	4	7	8	3	4	2	2	2	3	3	4

100 102 104 106 108 110 112 114 116 118 120 122

thus, for the first printf statement a, *a, **a give address of first element . since the indirection ***a gives the value. Hence, the first line of the output.
for the second printf a+1 increases in the third dimension thus points to value at 114, *a+1 increments in second dimension thus points to 104, **a +1 increments the first dimension thus points to 102 and ***a+1 first gets the value at first location and then increments it by 1. Hence, the output.

48) main( )

```
{
  int a[ ] = {10,20,30,40,50},j,*p;
  for(j=0; j<5; j++)
    {
      printf("%d" ,*a);
      a++;
    }
    p = a;
   for(j=0; j<5; j++)
      {
        printf("%d " ,*p);
        p++;
      }
 }
```

Answer:

Compiler error: lvalue required.

Explanation:

Error is in line with statement a++. The operand must be an lvalue and may be of any of scalar type for the any operator, array name only when subscripted is an lvalue. Simply array name is a non-modifiable lvalue.

49) main( )

```
{
 static int   a[ ]    = {0,1,2,3,4};
 int   *p[ ] = {a,a+1,a+2,a+3,a+4};
 int   **ptr =   p;
 ptr++;
 printf("\n %d   %d   %d", ptr-p, *ptr-a, **ptr);
```

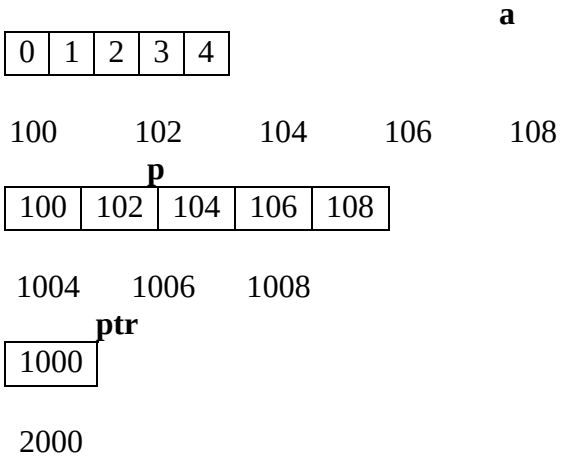
```
    *ptr++;
    printf("\n %d   %d   %d", ptr-p, *ptr-a, **ptr);
    *++ptr;
    printf("\n %d   %d   %d", ptr-p, *ptr-a, **ptr);
    ++*ptr;
printf("\n %d   %d   %d", ptr-p, *ptr-a, **ptr);
  }
```

Answer:

111
222
333
344

Explanation:

Let us consider the array and the two pointers with some address



After execution of the instruction ptr++ value in ptr becomes 1002, if scaling factor for integer is 2 bytes. Now ptr – p is value in ptr – starting location of array p, (1002 – 1000) / (scaling factor) = 1, *ptr – a = value at address pointed by ptr – starting value of array a, 1002 has a value 102 so the value is (102 – 100)/(scaling factor) = 1, **ptr is the value stored in the location pointed by the pointer of ptr = value pointed by value pointed by 1002 = value pointed by 102 = 1. Hence the output of the firs printf is 1, 1, 1.

After execution of *ptr++ increments value of the value in ptr by scaling factor, so it becomes1004. Hence, the outputs for the second printf are ptr – p = 2, *ptr – a = 2, **ptr = 2.

After execution of *++ptr increments value of the value in ptr by scaling factor, so it becomes1004. Hence, the outputs for the third printf are ptr – p = 3, *ptr – a = 3, **ptr = 3.

After execution of ++*ptr value in ptr remains the same, the value pointed by the value is incremented by the scaling factor. So the value in array p at location 1006 changes from 106 10 108,. Hence, the outputs for the fourth printf are ptr – p = 1006 – 1000 = 3, *ptr – a = 108 – 100 = 4, **ptr = 4.

50)
```
main( )
{
```

```
 char   *q;
 int   j;
 for (j=0; j<3; j++) scanf("%s" ,(q+j));
 for (j=0; j<3; j++) printf("%c" ,*(q+j));
 for (j=0; j<3; j++) printf("%s" ,(q+j));
}
```

Explanation:

Here we have only one pointer to type char and since we take input in the same pointer thus we keep writing over in the same location, each time shifting the pointer value by 1. Suppose the inputs are MOUSE, TRACK and VIRTUAL. Then for the first input suppose the pointer starts at location 100 then the input one is stored as

M	O	U	S	E	\0

When the second input is given the pointer is incremented as j value becomes 1, so the input is filled in memory starting from 101.

M	T	R	A	C	K	\0

The third input starts filling from the location 102

M	T	V	I	R	T	U	A	L	\0

This is the final value stored .
The first printf prints the values at the position q, q+1 and q+2 = M T V
The second printf prints three strings starting from locations q, q+1, q+2
i.e MTVIRTUAL, TVIRTUAL and VIRTUAL.

51)
```
main( )
{
 void *vp;
 char ch = 'g', *cp = "goofy";
 int j = 20;
 vp = &ch;
 printf("%c", *(char *)vp);
 vp = &j;
 printf("%d",*(int *)vp);
 vp = cp;
 printf("%s",(char *)vp + 3);
}
```

Answer:

g20fy

Explanation:

Since a void pointer is used it can be type casted to any other type pointer. vp = &ch stores address of char ch and the next statement prints the value stored in vp after type casting it to the proper data type pointer. the output is 'g'. Similarly the output from second printf is '20'. The third printf statement type casts it to

print the string from the 4th value hence the output is ‘fy’.

52) main ( )

```
{
 static char *s[ ]  = {“black”, “white”, “yellow”, “violet”};
 char **ptr[ ] = {s+3, s+2, s+1, s}, ***p;
 p = ptr;
 **++p;
 printf(“%s”,*--*++p + 3);
}
```

Answer:

ck

Explanation:

In this problem we have an array of char pointers pointing to start of 4 strings. Then we have ptr which is a pointer to a pointer of type char and a variable p which is a pointer to a pointer to a pointer of type char. p hold the initial value of ptr, i.e. p = s+3. The next statement increment value in p by 1 , thus now value of p = s+2. In the printf statement the expression is evaluated *++p causes gets value s+1 then the pre decrement is executed and we get s+1 – 1 = s . the indirection operator now gets the value from the array of s and adds 3 to the starting address. The string is printed starting from this position. Thus, the output is ‘ck’.

53) main()

```
{
 int  i, n;
 char *x = “girl”;
 n = strlen(x);
 *x = x[n];
 for(i=0; i<n; ++i)
   {
      printf(“%s\n”,x);
      x++;
   }
 }
```

Answer:

(blank space)
irl
rl
l

Explanation:

Here a string (a pointer to char) is initialized with a value “girl”. The strlen function returns the length of the string, thus n has a value 4. The next statement assigns value at the nth location (‘\0’) to the first location. Now the string becomes “\0irl” . Now the printf statement prints the string after each iteration it

increments it starting position. Loop starts from 0 to 4. The first time x[0] = '\0' hence it prints nothing and pointer value is incremented. The second time it prints from x[1] i.e "irl" and the third time it prints "rl" and the last time it prints "l" and the loop terminates.

54) int i,j;
```
 for(i=0;i<=10;i++)
 {
 j+=5;
 assert(i<5);
 }
```
Answer:

Runtime error: Abnormal program termination.
assert failed (i<5), <file name>,<line number>

Explanation:

asserts are used during debugging to make sure that certain conditions are satisfied. If assertion fails, the program will terminate reporting the same. After debugging use,

#undef NDEBUG

and this will disable all the assertions from the source code. Assertion is a good debugging tool to make use of.

55) main()
```
 {
 int i=-1;
 +i;
 printf("i = %d, +i = %d \n",i,+i);
 }
```
Answer:

i = -1, +i = -1

Explanation:

Unary + is the only dummy operator in C. Where-ever it comes you can just ignore it just because it has no effect in the expressions (hence the name dummy operator).

56) What are the files which are automatically opened when a C file is executed?

Answer:

stdin, stdout, stderr (standard input,standard output,standard error).

57) what will be the position of the file marker?
```
 a: fseek(ptr,0,SEEK_SET);
 b: fseek(ptr,0,SEEK_CUR);
```
Answer :

a: The SEEK_SET sets the file position marker to the starting of the file.
b: The SEEK_CUR sets the file position marker to the current position of the file.

58) main()
```
 {
 char name[10],s[12];
 scanf(" \"%[^\"]\"",s);
 }
```
How scanf will execute?

Answer:

First it checks for the leading white space and discards it.Then it matches with a quotation mark and then it reads all character upto another quotation mark.

59) What is the problem with the following code segment?
```
 while ((fgets(receiving array,50,file_ptr)) != EOF)
   ;
```
Answer & Explanation:

fgets returns a pointer. So the correct end of file check is checking for != NULL.

60) main()
```
 {
 main();
 }
```
Answer:

Runtime error : Stack overflow.

Explanation:

main function calls itself again and again. Each time the function is called its return address is stored in the call stack. Since there is no condition to terminate the function call, the call stack overflows at runtime. So it terminates the program and results in an error.

61) main()
```
 {
 char *cptr,c;
 void *vptr,v;
 c=10;   v=0;
 cptr=&c; vptr=&v;
 printf("%c%v",c,v);
 }
```
Answer:

Compiler error (at line number 4): size of v is Unknown.

Explanation:

You can create a variable of type void * but not of type void, since void is an empty type. In the second line you are creating variable vptr of type void * and v of type void hence an error.

62) main()
```
 {
```

```
char *str1="abcd";
char str2[]="abcd";
printf("%d %d %d",sizeof(str1),sizeof(str2),sizeof("abcd"));
}
```

Answer:

2 5 5

Explanation:

In first sizeof, str1 is a character pointer so it gives you the size of the pointer variable. In second sizeof the name str2 indicates the name of the array whose size is 5 (including the '\0' termination character). The third sizeof is similar to the second one.

63)
```
main()
{
char not;
not=!2;
printf("%d",not);
}
```

Answer:

0

Explanation:

! is a logical operator. In C the value 0 is considered to be the boolean value FALSE, and any non-zero value is considered to be the boolean value TRUE. Here 2 is a non-zero value so TRUE. !TRUE is FALSE (0) so it prints 0.

64)
```
#define FALSE -1
#define TRUE    1
#define NULL    0
main() {
    if(NULL)
  puts("NULL");
    else if(FALSE)
  puts("TRUE");
    else
  puts("FALSE");
    }
```

Answer:

TRUE

Explanation:

The input program to the compiler after processing by the preprocessor is,

```
 main(){
if(0)
  puts("NULL");
        else if(-1)
  puts("TRUE");
        else
```

```
  puts("FALSE");
 }
```

Preprocessor doesn't replace the values given inside the double quotes. The check by if condition is boolean value false so it goes to else. In second if -1 is boolean value true hence "TRUE" is printed.

65)
```
main()
 {
 int k=1;
 printf("%d==1 is ""%s",k,k==1?"TRUE":"FALSE");
 }
```

Answer:

1==1 is TRUE

Explanation:

When two strings are placed together (or separated by white-space) they are concatenated (this is called as "stringization" operation). So the string is as if it is given as "%d==1 is %s". The conditional operator(?:) evaluates to "TRUE".

66)
```
main()
 {
 int y;
 scanf("%d",&y); // input given is 2000
 if( (y%4==0 && y%100 != 0) || y%100 == 0 )
       printf("%d is a leap year");
 else
       printf("%d is not a leap year");
 }
```

Answer:

2000 is a leap year

Explanation:

An ordinary program to check if leap year or not.

67)
```
    #define max 5
 #define int arr1[max]
 main()
 {
 typedef char arr2[max];
 arr1 list={0,1,2,3,4};
 arr2 name="name";
 printf("%d %s",list[0],name);
 }
```

Answer:

Compiler error (in the line arr1 list = {0,1,2,3,4})

Explanation:

arr2 is declared of type array of size 5 of characters. So it can be used to declare the variable name of the type arr2. But it is not the case of arr1. Hence an error.

Rule of Thumb:

#defines are used for textual replacement whereas typedefs are used for declaring new types.

```
68) int i=10;
 main()
 {
   extern int i;
                {
       int i=20;
  {
   const volatile unsigned i=30;
   printf("%d",i);
  }
         printf("%d",i);
     }
 printf("%d",i);
 }
```

Answer:

30,20,10

Explanation:

'{' introduces new block and thus new scope. In the innermost block i is declared as,

const volatile unsigned

which is a valid declaration. i is assumed of type int. So printf prints 30. In the next block, i has value 20 and so printf prints 20. In the outermost block, i is declared as extern, so no storage space is allocated for it. After compilation is over the linker resolves it to global variable i (since it is the only variable visible there). So it prints i's value as 10.

```
69) main()
 {
      int *j;
      {
       int i=10;
       j=&i;
       }
       printf("%d",*j);
         }
```

Answer:

10

Explanation:

The variable i is a block level variable and the visibility is inside that block only. But the lifetime of i is lifetime of the function so it lives upto the exit of main function. Since the i is still allocated space, *j prints the value stored in i since j points i.

70) main()
```
 {
 int i=-1;
 -i;
 printf("i = %d, -i = %d \n",i,-i);
 }
```
Answer:

i = -1, -i = 1

Explanation:

-i is executed and this execution doesn't affect the value of i. In printf first you just print the value of i. After that the value of the expression -i = -(-1) is printed.

71) #include<stdio.h>
```
main()
 {
   const int i=4;
   float j;
   j = ++i;
   printf("%d   %f", i,++j);
 }
```
Answer:

Compiler error

Explanation:

i is a constant. you cannot change the value of constant

72) #include<stdio.h>
```
main()
{
  int a[2][2][2] = { {10,2,3,4}, {5,6,7,8}   };
  int *p,*q;
  p=&a[2][2][2];
  *q=***a;
  printf("%d..%d",*p,*q);
}
```
Answer:

garbagevalue..1

Explanation:

p=&a[2][2][2] you declare only two 2D arrays. but you are trying to access the third 2D(which you are not declared) it will print garbage values. *q=***a starting address of a is assigned integer pointer. now q is pointing to starting address of a.if you print *q meAnswer:it will print first element of 3D array.

73) #include<stdio.h>
```
main()
  {
```

```
        register i=5;
        char j[]= "hello";
         printf("%s   %d",j,i);
    }
```

Answer:

hello 5

Explanation:

if you declare i as register compiler will treat it as ordinary integer and it will take integer value. i value may be stored either in register or in memory.

74)
```
main()
    {
  int i=5,j=6,z;
  printf("%d",i+++j);
 }
```

Answer:

11

Explanation:

the expression i+++j is treated as (i++ + j)

76)
```
struct aaa{
        struct aaa *prev;
        int i;
        struct aaa *next;
        };
    main()
    {
     struct aaa abc,def,ghi,jkl;
     int x=100;
     abc.i=0;abc.prev=&jkl;
     abc.next=&def;
     def.i=1;def.prev=&abc;def.next=&ghi;
     ghi.i=2;ghi.prev=&def;
     ghi.next=&jkl;
     jkl.i=3;jkl.prev=&ghi;jkl.next=&abc;
     x=abc.next->next->prev->next->i;
     printf("%d",x);
    }
```

Answer:

2

Explanation:

above all statements form a double circular linked list;

abc.next->next->prev->next->i

this one points to "ghi" node the value of at particular node is 2.

77)
```
struct point
```

```
 {
 int x;
 int y;
 };
struct point origin,*pp;
main()
{
pp=&origin;
printf("origin is(%d%d)\n",(*pp).x,(*pp).y);
printf("origin is (%d%d)\n",pp->x,pp->y);
}
```

Answer:

origin is(0,0)
origin is(0,0)

Explanation:

pp is a pointer to structure. we can access the elements of the structure either with arrow mark or with indirection operator.

Note:

Since structure point is globally declared x & y are initialized as zeroes

78)
```
main()
{
 int i=_l_abc(10);
printf("%d\n",--i);
}
int _l_abc(int i)
{
 return(i++);
}
```

Answer:

9

Explanation:

return(i++) it will first return i and then increments. i.e. 10 will be returned.

79)
```
main()
{
 char *p;
 int *q;
 long *r;
 p=q=r=0;
 p++;
 q++;
 r++;
 printf("%p...%p...%p",p,q,r);
}
```

Answer:

0001...0002...0004

Explanation:

++ operator when applied to pointers increments address according to their corresponding data-types.

80) main()

```
{
 char c=' ',x,convert(z);
 getc(c);
 if((c>='a') && (c<='z'))
 x=convert(c);
 printf("%c",x);
}
convert(z)
{
  return z-32;
}
```

Answer:

Compiler error

Explanation:

declaration of convert and format of getc() are wrong.

81) main(int argc, char **argv)

```
{
 printf("enter the character");
 getchar();
 sum(argv[1],argv[2]);
}
sum(num1,num2)
int num1,num2;
{
 return num1+num2;
}
```

Answer:

Compiler error.

Explanation:

argv[1] & argv[2] are strings. They are passed to the function sum without converting it to integer values.

82) # include <stdio.h>

```
int one_d[]={1,2,3};
main()
{
 int *ptr;
 ptr=one_d;
```

```
 ptr+=3;
 printf("%d",*ptr);
}
```

Answer:

garbage value

Explanation:

ptr pointer is pointing to out of the array range of one_d.

83) # include<stdio.h>

```
aaa() {
  printf("hi");
 }
bbb(){
 printf("hello");
 }
ccc(){
 printf("bye");
 }
main()
{
  int (*ptr[3])();
  ptr[0]=aaa;
  ptr[1]=bbb;
  ptr[2]=ccc;
  ptr[2]();
}
```

Answer:

bye

Explanation:

ptr is array of pointers to functions of return type int.ptr[0] is assigned to address of the function aaa. Similarly ptr[1] and ptr[2] for bbb and ccc respectively. ptr[2]() is in effect of writing ccc(), since ptr[2] points to ccc.

85) #include<stdio.h>

```
main()
{
FILE *ptr;
char i;
ptr=fopen("zzz.c","r");
while((i=fgetch(ptr))!=EOF)
        printf("%c",i);
}
```

Answer:

contents of zzz.c followed by an infinite loop

Explanation:

The condition is checked against EOF, it should be checked against NULL.

86) main()
```
{
 int i =0;j=0;
 if(i && j++)
     printf("%d..%d",i++,j);
printf("%d..%d,i,j);
}
```
Answer:

0..0

Explanation:

The value of i is 0. Since this information is enough to determine the truth value of the boolean expression. So the statement following the if statement is not executed. The values of i and j remain unchanged and get printed.

87) main()
```
{
 int i;
 i = abc();
 printf("%d",i);
}
abc()
{
 _AX = 1000;
}
```
Answer:

1000

Explanation:

Normally the return value from the function is through the information from the accumulator. Here _AH is the pseudo global variable denoting the accumulator. Hence, the value of the accumulator is set 1000 so the function returns value 1000.

88) int i;
```
  main(){
int t;
for ( t=4;scanf("%d",&i)-t;printf("%d\n",i))
                printf("%d--",t--);
        }
```
// If the inputs are 0,1,2,3 find the o/p

Answer:

```
  4--0
3--1
2--2
```
Explanation:

Let us assume some x= scanf("%d",&i)-t the values during execution

will be,

t	i	x
4	0	-4
3	1	-2
2	2	0

89)
```
main(){
    int a= 0;int b = 20;char x =1;char y =10;
    if(a,b,x,y)
        printf("hello");
}
```
Answer:

hello

Explanation:

The comma operator has associativity from left to right. Only the rightmost value is returned and the other values are evaluated and ignored. Thus the value of last variable y is returned to check in if. Since it is a non zero value if becomes true so, "hello" will be printed.

90)
```
main(){
    unsigned int i;
    for(i=1;i>-2;i--)
        printf("c aptitude");
}
```
Explanation:

i is an unsigned integer. It is compared with a signed value. Since the both types doesn't match, signed is promoted to unsigned value. The unsigned equivalent of -2 is a huge value so condition becomes false and control comes out of the loop.

91) In the following pgm add a stmt in the function fun such that the address of 'a' gets stored in 'j'.
```
main(){
    int * j;
    void fun(int **);
    fun(&j);
}
void fun(int **k) {
    int a =0;
    /* add a stmt here*/
}
```
Answer:

*k = &a

Explanation:

The argument of the function is a pointer to a pointer.

92) What are the following notations of defining functions known as?

i. int abc(int a,float b)
{
/* some code */
}

ii. int abc(a,b)
int a; float b;
{
/* some code*/
}

Answer:

i. ANSI C notation
ii. Kernighan & Ritche notation

93) main()
{
char *p;
p="%d\n";
p++;
p++;
printf(p-2,300);
}

Answer:

300

Explanation:

The pointer points to % since it is incremented twice and again decremented by 2, it points to '%d\n' and 300 is printed.

94) main(){
char a[100];
a[0]='a';a[1]]='b';a[2]='c';a[4]='d';
abc(a);
}
abc(char a[]){
a++;
printf("%c",*a);
a++;
printf("%c",*a);
}

Explanation:

The base address is modified only in function and as a result a points to 'b' then after incrementing to 'c' so bc will be printed.

95) func(a,b)
int a,b;
{
return(a= (a==b));

```
}
main()
{
int process(),func();
printf("The value of process is %d !\n ",process(func,3,6));
}
process(pf,val1,val2)
int (*pf) ();
int val1,val2;
{
return((*pf) (val1,val2));
 }
```

Answer:

The value if process is 0 !

Explanation:

The function 'process' has 3 parameters - 1, a pointer to another function 2 and 3, integers. When this function is invoked from main, the following substitutions for formal parameters take place: func for pf, 3 for val1 and 6 for val2. This function returns the result of the operation performed by the function 'func'. The function func has two integer parameters. The formal parameters are substituted as 3 for a and 6 for b. since 3 is not equal to 6, a==b returns 0. therefore the function returns 0 which in turn is returned by the function 'process'.

96)

```
void main()
{
 static int i=5;
 if(--i){
   main();
   printf("%d ",i);
 }
}
```

Answer:

0 0 0 0

Explanation:

The variable "I" is declared as static, hence memory for I will be allocated for only once, as it encounters the statement. The function main() will be called recursively unless I becomes equal to 0, and since main() is recursively called, so the value of static I ie., 0 will be printed every time the control is returned.

97)

```
void main()
{
 int k=ret(sizeof(float));
 printf("\n here value is %d",++k);
}
int ret(int ret)
{
```

```
 ret += 2.5;
 return(ret);
}
```

Answer:

Here value is 7

Explanation:

The int ret(int ret), ie., the function name and the argument name can be the same.

Firstly, the function ret() is called in which the sizeof(float) ie., 4 is passed, after the first expression the value in ret will be 6, as ret is integer hence the value stored in ret will have implicit type conversion from float to int. The ret is returned in main() it is printed after and preincrement.

98)

```
void main()
{
 char a[]="12345\0";
 int i=strlen(a);
 printf("here in 3 %d\n",++i);
}
```

Answer:

here in 3 6

Explanation:

The char array 'a' will hold the initialized string, whose length will be counted from 0 till the null character. Hence the 'I' will hold the value equal to 5, after the pre-increment in the printf statement, the 6 will be printed.

99)

```
void main()
{
 unsigned giveit=-1;
 int gotit;
 printf("%u ",++giveit);
 printf("%u \n",gotit=--giveit);
}
```

Answer:

0 65535

Explanation:

100)

```
void main()
{
 int i;
 char a[]="\0";
 if(printf("%s\n",a))
   printf("Ok here \n");
 else
   printf("Forget it\n");
}
```

Answer:

Ok here

Explanation:

Printf will return how many characters does it print. Hence printing a null character returns 1 which makes the if statement true, thus "Ok here" is printed.

101) void main()

```
{
 void *v;
 int integer=2;
 int *i=&integer;
 v=i;
 printf("%d",(int*)*v);
}
```

Answer:

Compiler Error. We cannot apply indirection on type void*.

Explanation:

Void pointer is a generic pointer type. No pointer arithmetic can be done on it. Void pointers are normally used for,

1. Passing generic pointers to functions and returning such pointers.
2. As a intermediate pointer type.
3. Used when the exact pointer type will be known at a later point of time.

102) void main()

```
{
 int i=i++,j=j++,k=k++;
        printf(“%d%d%d”,i,j,k);
}
```

Answer:

Garbage values.

Explanation:

An identifier is available to use in program code from the point of its declaration. So expressions such as i = i++ are valid statements. The i, j and k are automatic variables and so they contain some garbage value. *Garbage in is garbage out (GIGO).*

103) void main()

```
{
 static int i=i++, j=j++, k=k++;
        printf(“i = %d j = %d k = %d”, i, j, k);
}
```

Answer:

i = 1 j = 1 k = 1

Explanation:

Since static variables are initialized to zero by default.

104) void main()
```
{
 while(1){
  if(printf("%d",printf("%d")))
   break;
  else
   continue;
 }
}
```
Answer:

Garbage values

Explanation:

The inner printf executes first to print some garbage value. The printf returns no of characters printed and this value also cannot be predicted. Still the outer printf prints something and so returns a non-zero value. So it encounters the break statement and comes out of the while statement.

104) main()
```
{
 unsigned int i=10;
 while(i-->=0)
  printf("%u ",i);

}
```
Answer:

10 9 8 7 6 5 4 3 2 1 0 65535 65534…..

Explanation:

Since i is an unsigned integer it can never become negative. So the expression i-->=0 will always be true, leading to an infinite loop.

105) #include<conio.h>
```
main()
{
 int x,y=2,z,a;
 if(x=y%2) z=2;
 a=2;
 printf("%d %d ",z,x);
}
```
Answer:

Garbage-value 0

Explanation:

The value of y%2 is 0. This value is assigned to x. The condition reduces to if (x) or in other words if(0) and so z goes uninitialized.

Thumb Rule: **C**heck all control paths to write bug free code.

106) main()
{
 int a[10];
 printf("%d",*a+1-*a+3);
}
Answer:
4
Explanation:
*a and -*a cancels out. The result is as simple as 1 + 3 = 4 !

107) #define prod(a,b) a*b
main()
{
 int x=3,y=4;
 printf("%d",prod(x+2,y-1));
}
Answer:
10
Explanation:
The macro expands and evaluates to as:
x+2*y-1 => x+(2*y)-1 => 10

108) main()
{
 unsigned int i=65000;
 while(i++!=0);
 printf("%d",i);
}
Answer:
1
Explanation:
Note the semicolon after the while statement. When the value of i becomes 0 it comes out of while loop. Due to post-increment on i the value of i while printing is 1.

109) main()
{
 int i=0;
 while(+(+i--)!=0)
  i-=i++;
 printf("%d",i);
}
Answer:
-1
Explanation:

Unary + is the only dummy operator in C. So it has no effect on the expression and now the while loop is, while(i--!=0) which is false and so breaks out of while loop. The value –1 is printed due to the post-decrement operator.

113) main()
```
{
 float f=5,g=10;
 enum{i=10,j=20,k=50};
 printf("%d\n",++k);
 printf("%f\n",f<<2);
 printf("%lf\n",f%g);
 printf("%lf\n",fmod(f,g));
}
```
Answer:

Line no 5: Error: Lvalue required
Line no 6: Cannot apply leftshift to float
Line no 7: Cannot apply mod to float

Explanation:

Enumeration constants cannot be modified, so you cannot apply ++.
Bit-wise operators and % operators cannot be applied on float values.
fmod() is to find the modulus values for floats as % operator is for ints.

110)
```
main()
{
 int i=10;
 void pascal f(int,int,int);
        f(i++,i++,i++);
 printf(" %d",i);
}
void pascal f(integer :i,integer:j,integer :k)
{
        write(i,j,k);
}
```
Answer:

Compiler error: unknown type integer
Compiler error: undeclared function write

Explanation:

Pascal keyword doesn't mean that pascal code can be used. It means that the function follows Pascal argument passing mechanism in calling the functions.

111)
```
void pascal f(int i,int j,int k)
{
        printf("%d %d %d",i, j, k);
}
void cdecl f(int i,int j,int k)
{
```

```
        printf("%d %d %d",i, j, k);
}
main()
{
 int i=10;
        f(i++,i++,i++);
 printf(" %d\n",i);
        i=10;
        f(i++,i++,i++);
        printf(" %d",i);
}
```

Answer:

10 11 12 13
12 11 10 13

Explanation:

Pascal argument passing mechanism forces the arguments to be called from left to right. cdecl is the normal C argument passing mechanism where the arguments are passed from right to left.

112). What is the output of the program given below

```
main()
    {
       signed char i=0;
       for(;i>=0;i++) ;
       printf("%d\n",i);
    }
```

Answer

-128

Explanation

Notice the semicolon at the end of the for loop. THe initial value of the i is set to 0. The inner loop executes to increment the value from 0 to 127 (the positive range of char) and then it rotates to the negative value of -128. The condition in the for loop fails and so comes out of the for loop. It prints the current value of i that is -128.

113)

```
main()
    {
       unsigned char i=0;
       for(;i>=0;i++) ;
       printf("%d\n",i);
    }
```

Answer

infinite loop

Explanation

The difference between the previous question and this one is that the char is

declared to be unsigned. So the i++ can never yield negative value and i>=0 never becomes false so that it can come out of the for loop.

114) main()
```
{
        char i=0;
        for(;i>=0;i++) ;
        printf("%d\n",i);

 }
```
Answer:

Behavior is implementation dependent.

Explanation:

The detail if the char is signed/unsigned by default is implementation dependent. If the implementation treats the char to be signed by default the program will print –128 and terminate. On the other hand if it considers char to be unsigned by default, it goes to infinite loop.

Rule:

You can write programs that have implementation dependent behavior. But dont write programs that depend on such behavior.

115) Is the following statement a declaration/definition. Find what does it mean?

int (*x)[10];

Answer

Definition.

x is a pointer to array of(size 10) integers.

Apply clock-wise rule to find the meaning of this definition.

116). What is the output for the program given below

```
typedef enum errorType{warning, error, exception,}error;
main()
{
    error g1;
    g1=1;
    printf("%d",g1);
}
```
Answer

Compiler error: Multiple declaration for error

Explanation

The name error is used in the two meanings. One means that it is a enumerator constant with value 1. The another use is that it is a type name (due to typedef) for enum errorType. Given a situation the compiler cannot distinguish the meaning of error to know in what sense the error is used:

```
        error g1;
            g1=error;
 // which error it refers in each case?
```

When the compiler can distinguish between usages then it will not issue error (in pure technical terms, names can only be overloaded in different namespaces).

Note: the extra comma in the declaration,

```
enum errorType{warning, error, exception,}
```

is not an error. An extra comma is valid and is provided just for programmer's convenience.

117)

```
typedef struct error{int warning, error, exception;}error;
main()
{
    error g1;
    g1.error =1;
    printf("%d",g1.error);
}
```

Answer

1

Explanation

The three usages of name errors can be distinguishable by the compiler at any instance, so valid (they are in different namespaces).

```
Typedef struct error{int warning, error, exception;}error;
```

This error can be used only by preceding the error by struct kayword as in:

```
struct error someError;
typedef struct error{int warning, error, exception;}error;
```

This can be used only after . (dot) or -> (arrow) operator preceded by the variable name as in :

```
g1.error =1;
  printf("%d",g1.error);
typedef struct error{int warning, error, exception;}error;
```

This can be used to define variables without using the preceding struct keyword as in:

```
error g1;
```

Since the compiler can perfectly distinguish between these three usages, it is perfectly legal and valid.

Note

This code is given here to just explain the concept behind. In real programming don't use such overloading of names. It reduces the readability of the code. Possible doesn't mean that we should use it!

118)

```
#ifdef something
    int some=0;
```

```
#endif

main()
{
        int thing = 0;
        printf("%d %d\n", some ,thing);
}
```

Answer:

Compiler error : undefined symbol some

Explanation:

This is a very simple example for conditional compilation. The name something is not already known to the compiler making the declaration
int some = 0;
effectively removed from the source code.

119)
```
#if something == 0
int some=0;
#endif

main()
{
        int thing = 0;
        printf("%d %d\n", some ,thing);
}
```

Answer

0 0

Explanation

This code is to show that preprocessor expressions are not the same as the ordinary expressions. If a name is not known the preprocessor treats it to be equal to zero.

120). What is the output for the following program

```
main()
        {
    int arr2D[3][3];
     printf("%d\n", ((arr2D==* arr2D)&&(* arr2D == arr2D[0])) );
      }
```

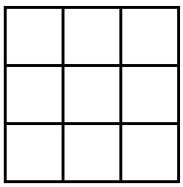
Answer

1

Explanation

This is due to the close relation between the arrays and pointers. N dimensional arrays are made up of (N-1) dimensional arrays.

arr2D is made up of a 3 single arrays that contains 3 integers each .



The name arr2D refers to the beginning of all the 3 arrays. *arr2D refers to the start of the first 1D array (of 3 integers) that is the same address as arr2D. So the expression (arr2D == *arr2D) is true (1).
Similarly, *arr2D is nothing but *(arr2D + 0), adding a zero doesn't change the value/meaning. Again arr2D[0] is the another way of telling *(arr2D + 0). So the expression (*(arr2D + 0) == arr2D[0]) is true (1).
Since both parts of the expression evaluates to true the result is true(1) and the same is printed.

121) void main()
{
if(~0 == (unsigned int)-1)
printf("You can answer this if you know how values are represented in memory");
}
Answer
You can answer this if you know how values are represented in memory
Explanation
~ (tilde operator or bit-wise negation operator) operates on 0 to produce all ones to fill the space for an integer. –1 is represented in unsigned value as all 1's and so both are equal.

122) int swap(int *a,int *b)
{
*a=*a+*b;*b=*a-*b;*a=*a-*b;
}
main()
{
int x=10,y=20;
swap(&x,&y);
printf("x= %d y = %d\n",x,y);
}
Answer
x = 20 y = 10
Explanation

This is one way of swapping two values. Simple checking will help understand this.

123) main()
```
{
char *p = “ayqm”;
printf(“%c”,++*(p++));
}
```
Answer:
b

124) main()
```
{
  int i=5;
  printf("%d",++i++);
}
```
Answer:
Compiler error: Lvalue required in function main

Explanation:
++i yields an rvalue. For postfix ++ to operate an lvalue is required.

125) main()
```
{
    char *p = “ayqm”;
    char c;
    c = ++*p++;
    printf(“%c”,c);
}
```
Answer:
b

Explanation:
There is no difference between the expression ++*(p++) and ++*p++. Parenthesis just works as a visual clue for the reader to see which expression is first evaluated.

126)
```
int aaa() {printf(“Hi”);}
int bbb(){printf(“hello”);}
iny ccc(){printf(“bye”);}

main()
{
int ( * ptr[3]) ();
ptr[0] = aaa;
ptr[1] = bbb;
ptr[2] =ccc;
```

```
        ptr[2]();
}
```
Answer:

bye

Explanation:

int (* ptr[3])() says that ptr is an array of pointers to functions that takes no arguments and returns the type int. By the assignment ptr[0] = aaa; it means that the first function pointer in the array is initialized with the address of the function aaa. Similarly, the other two array elements also get initialized with the addresses of the functions bbb and ccc. Since ptr[2] contains the address of the function ccc, the call to the function ptr[2]() is same as calling ccc(). So it results in printing "bye".

127)

```
main()
{
int i=5;
printf("%d",i=++i ==6);
}
```

Answer:

1

Explanation:

The expression can be treated as i = (++i==6), because == is of higher precedence than = operator. In the inner expression, ++i is equal to 6 yielding true(1). Hence the result.

128)

```
main()
{
  char p[ ]="%d\n";
        p[1] = 'c';
        printf(p,65);
}
```

Answer:

A

Explanation:

Due to the assignment p[1] = 'c' the string becomes, "%c\n". Since this string becomes the format string for printf and ASCII value of 65 is 'A', the same gets printed.

129)

```
void ( * abc( int, void ( *def) () ) ) ();
```

Answer::

abc is a ptr to a function which takes 2 parameters .(a). an integer variable.(b). a ptrto a funtion which returns void. the return type of the function is void.

Explanation:

Apply the clock-wise rule to find the result.

130) main()
{
while (strcmp(“some”,”some\0”))
printf(“Strings are not equal\n”);
}

Answer:

No output

Explanation:

Ending the string constant with \0 explicitly makes no difference. So “some” and “some\0” are equivalent. So, strcmp returns 0 (false) hence breaking out of the while loop.

131) main()
{
char str1[] = {‘s’,’o’,’m’,’e’};
char str2[] = {‘s’,’o’,’m’,’e’,’\0’};
while (strcmp(str1,str2))
printf(“Strings are not equal\n”);
}

Answer:

“Strings are not equal”
“Strings are not equal”
….

Explanation:

If a string constant is initialized explicitly with characters, ‘\0’ is not appended automatically to the string. Since str1 doesn’t have null termination, it treats whatever the values that are in the following positions as part of the string until it randomly reaches a ‘\0’. So str1 and str2 are not the same, hence the result.

132) main()
{
int i = 3;
for (;i++=0;) printf(“%d”,i);
}

Answer:

Compiler Error: Lvalue required.

Explanation:

As we know that increment operators return rvalues and hence it cannot appear on the left hand side of an assignment operation.

133) void main()
{

```
        int *mptr, *cptr;
        mptr = (int*)malloc(sizeof(int));
        printf("%d",*mptr);
        int *cptr = (int*)calloc(sizeof(int),1);
        printf("%d",*cptr);
}
```

Answer:

garbage-value 0

Explanation:

The memory space allocated by malloc is uninitialized, whereas calloc returns the allocated memory space initialized to zeros.

134)

```
void main()
{
        static int i;
        while(i<=10)
        (i>2)?i++:i--;
 printf("%d", i);
}
```

Answer:

32767

Explanation:

Since i is static it is initialized to 0. Inside the while loop the conditional operator evaluates to false, executing i--. This continues till the integer value rotates to positive value (32767). The while condition becomes false and hence, comes out of the while loop, printing the i value.

135)

```
main()
{
int i=10,j=20;
      j = i, j?(i,j)?i:j:j;
printf("%d %d",i,j);
}
```

Answer:

10 10

Explanation:

The Ternary operator (? :) is equivalent for if-then-else statement. So the question can be written as:

```
if(i,j)
    {
        if(i,j)
    j = i;
else
   j = j;
}
```

else
j = j;

136) 1. const char *a;
2. char* const a;
3. char const *a;
-Differentiate the above declarations.

Answer:

1. 'const' applies to char * rather than 'a' (pointer to a constant char)
*a='F' : illegal *14. What are the processes that are not bothered by the swapper? Give Reason.*
a="Hi" : legal Zombie process: They do not take any up physical memory.
Processes locked in memories that are updating the region of the process.

2. 'const' applies to 'a' rather than to the value of a (constant pointer to char) Kernel swaps only the sleeping processes rather than the 'ready-to-run' processes, as they have the higher probability of being scheduled than the Sleeping processes.
*a='F' : legal
a="Hi" : illegal *15. What are the requirements for a swapper to work?*

The swapper works on the highest scheduling priority. Firstly it will look for any sleeping process, if not found then it will look for the ready-to-run process for swapping. But the major requirement for the swapper to work the ready-to-run process must be core-resident for at least 2 seconds before swapping out. And for swapping in the process must have been resided in the swap device for at least 2 seconds. If the requirement is not satisfied then the swapper will go into the wait state on that event and it is awaken once in a second by the Kernel.

3. Same as 1.

16. What are the criteria for choosing a process for swapping into memory from the swap device?

137) main() The resident time of the processes in the swap device, the priority of the processes and the amount of time the processes had been swapped out.
{
int i=5,j=10; *17. What are the criteria for choosing a process for swapping out of the memory to the swap device?*
i=i&=j&&10; The process's memory resident time,
printf("%d %d",i,j); Priority of the process and
} The nice value.

Answer: *18. What do you mean by nice value?*

1 10 Nice value is the value that controls {increments or decrements} the priority of the process. This value that is returned by the nice () system call. The equation for using nice value is:

Explanation: *Priority = ("recent CPU usage"/constant) + (base- priority) + (nice value)*

The expression can be written as i=(i&=(j&&10)); The inner expression (j&&10) evaluates to 1 because j==10. i is 5. i = 5&1 is 1. Hence the result. Only the

administrator can supply the nice value. The nice () system call works for the running process only. Nice value of one process cannot affect the nice value of the

138) main()
{
int i=4,j=7;
j = j || i++ && printf("YOU CAN");
printf("%d %d", i, j);
4 1
The boolean expression needs to be evaluated only till the truth value of the expression is not known. j is not equal to zero itself means that the expression's truth value is 1. Because it is followed by || and *true || (anything) => true where (anything) will not be evaluated.* So the remaining expression is not evaluated and so the value of i remains the same.
Similarly when && operator is involved in an expression, when any of the operands become false, the whole expression's truth value becomes false and hence the remaining expression will not be evaluated.
false && (anything) => false where (anything) will not be evaluated.

139) main()
{
register int a=2;
printf("Address of a = %d",&a);
printf("Value of a = %d",a);
}
Answer:
Compier Error: '&' on register variable
Rule to Remember:
& (address of) operator cannot be applied on register variables.

140) main()
{
float i=1.5;
switch(i)
{
case 1: printf("1");
case 2: printf("2");
default : printf("0");
}
}
Answer:
Compiler Error: switch expression not integral
Explanation:
Switch statements can be applied only to integral types.

```
141) main()
        {
  extern i;
         printf("%d\n",i);

    }
```

There is no space in the main memory for the new incoming process.
All processes in the main memory are asleep.
All 'ready-to-run' processes are swapped out.
There is no space in the swap device for the new incoming process that are swapped out of the main memory.

20. What are conditions for a machine to support Demand Paging?

Answer: Memory architecture must based on Pages,
The machine must support the 'restartable' instructions.

other process.

19. What are conditions on which deadlock can occur while swapping the processes?

21. What is 'the principle of locality'?

It's the nature of the processes that they refer only to the small subset of the total data space of the process. i.e. the process frequently calls the same subroutines or executes the loop instructions.

22. What is the working set of a process?

The set of pages that are referred by the process in the last 'n', references, where 'n' is called the *window* of the working set of the process.

23. What is the window of the working set of a process?

The window of the working set of a process is the total number in which the process had referred the set of pages in the working set of the process.

24. What is called a page fault?

Page fault is referred to the situation when the process addresses a page in the working set of the process but the process fails to locate the page in the working set. And on a page fault the kernel updates the working set by reading the page from the secondary device.

25. What are data structures that are used for Demand Paging?

Explanation:

```
{
 int i=20;
        printf("%d\n",i);
```

```
}
        }
```

Answer:

Linker Error : Unresolved external symbol i

Explanation:

The identifier i is available in the inner block and so using extern has no use in resolving it.

```
142) main()
        {
  int a=2,*f1,*f2;
         f1=f2=&a;
  *f2+=*f2+=a+=2.5;
         printf("\n%d %d %d",a,*f1,*f2);
        }
```

Answer:

16 16 16

Explanation:

f1 and f2 both refer to the same memory location a. So changes through f1 and f2 ultimately affects only the value of a.

```
143) main()
        {
  char *p="GOOD";
         char a[ ]="GOOD";
                printf("\n sizeof(p) = %d, sizeof(*p) = %d, strlen(p) = %d", sizeof(p), sizeof(*p),
                strlen(p));
         printf("\n sizeof(a) = %d, strlen(a) = %d", sizeof(a), strlen(a));
        }
```

Answer:

```
  sizeof(p) = 2, sizeof(*p) = 1, strlen(p) = 4
         sizeof(a) = 5, strlen(a) = 4
```

Explanation:

```
  sizeof(p) => sizeof(char*) => 2
         sizeof(*p) => sizeof(char) => 1
  Similarly,
         sizeof(a) => size of the character array => 5
```

When sizeof operator is applied to an array it returns the sizeof the array and it is not the same as the sizeof the pointer variable. Here the sizeof(a) where a is the character array and the size of the array is 5 because the space necessary for the terminating NULL character should also be taken into account.

```
144) #define DIM( array, type) sizeof(array)/sizeof(type)
        main()
        {
                int arr[10];
```

```
        printf("The dimension of the array is %d", DIM(arr, int));
}
```

Answer:

10

Explanation:

The size of integer array of 10 elements is 10 * sizeof(int). The macro expands to sizeof(arr)/sizeof(int) => 10 * sizeof(int) / sizeof(int) => 10.

145)
```
int DIM(int array[])
{
return sizeof(array)/sizeof(int );
}
main()
{
        int arr[10];
        printf("The dimension of the array is %d", DIM(arr));
}
```

Answer:

1

Explanation:

Arrays cannot be passed to functions as arguments and only the pointers can be passed. So the argument is equivalent to int * array (this is one of the very few places where [] and * usage are equivalent). The return statement becomes, sizeof(int *)/ sizeof(int) that happens to be equal in this case.

146)
```
main()
{
 static int a[3][3]={1,2,3,4,5,6,7,8,9};
 int i,j;
 static *p[]={a,a+1,a+2};
for(i=0;i<3;i++)
 {
 for(j=0;j<3;j++)
 printf("%d\t%d\t%d\t%d\n",*(*(p+i)+j),
 *(*(j+p)+i),*(*(i+p)+j),*(*(p+j)+i));
}
}
```

Answer:

```
1       1       1       1
2       4       2       4
        3       7       3       7
4       2       4       2
5       5       5       5
        6       8       6       8
7       3       7       3
8       6       8       6
```

9 9 9 9

Explanation:

((p+i)+j) is equivalent to p[i][j].

147) main()
{
void swap();
int x=10,y=8;
swap(&x,&y);
printf("x=%d y=%d",x,y);
}
void swap(int *a, int *b)
{
*a ^= *b, *b ^= *a, *a ^= *b;
}

Answer:

x=10 y=8

Explanation:

Using ^ like this is a way to swap two variables without using a temporary variable and that too in a single statement.

Inside main(), void swap(); means that swap is a function that may take any number of arguments (not no arguments) and returns nothing. So this doesn't issue a compiler error by the call swap(&x,&y); that has two arguments.

This convention is historically due to pre-ANSI style (referred to as Kernighan and Ritchie style) style of function declaration. In that style, the swap function will be defined as follows,

```
void swap()
int *a, int *b
{
    *a ^= *b,   *b ^= *a, *a ^= *b;
}
```

where the arguments follow the (). So naturally the declaration for swap will look like, void swap() which means the swap can take any number of arguments.

148) main()
{
int i = 257;
int *iPtr = &i;
printf("%d %d", *((char*)iPtr), *((char*)iPtr+1));
}

Answer:

1 1

Explanation:

The integer value 257 is stored in the memory as, 00000001 00000001, so the individual bytes are taken by casting it to char * and get printed.

149) main()
{
int i = 258;
int *iPtr = &i;
printf("%d %d", *((char*)iPtr), *((char*)iPtr+1));
}

Answer:

2 1

Explanation:

The integer value 257 can be represented in binary as, 00000001 00000001. Remember that the INTEL machines are 'small-endian' machines. *Small-endian means that the lower order bytes are stored in the higher memory addresses and the higher order bytes are stored in lower addresses*. The integer value 258 is stored in memory as: 00000001 00000010.

150) main()
{
int i=300;
char *ptr = &i;
*++ptr=2;
printf("%d",i);
}

Answer:

556

Explanation:

The integer value 300 in binary notation is: 00000001 00101100. It is stored in memory (small-endian) as: 00101100 00000001. Result of the expression *++ptr = 2 makes the memory representation as: 00101100 00000010. So the integer corresponding to it is 00000010 00101100 => 556.

151) #include <stdio.h>
main()
{
char * str = "hello";
char * ptr = str;
char least = 127;
while (*ptr++)
least = (*ptr<least) ?*ptr :least;
printf("%d",least);
}

Answer:

0

Explanation:

After 'ptr' reaches the end of the string the value pointed by 'str' is '\0'. So the value of 'str' is less than that of 'least'. So the value of 'least' finally is 0.

152) Declare an array of N pointers to functions returning pointers to functions returning pointers to characters?

Answer:

```
(char*(*)( )) (*ptr[N])( );
```

153)
```
main()
{
        struct student
        {
                char name[30];
                struct date dob;
        }stud;
        struct date
                 {
                  int day,month,year;
                  };
              scanf("%s%d%d%d", stud.rollno, &student.dob.day, &student.dob.month,
        &student.dob.year);
}
```

Answer:

Compiler Error: Undefined structure date

Explanation:

Inside the struct definition of 'student' the member of type struct date is given. The compiler doesn't have the definition of date structure (forward reference is not allowed in C in this case) so it issues an error.

154)
```
main()
{
        struct date;
        struct student
{
        char name[30];
        struct date dob;
}stud;
        struct date
          {
                  int day,month,year;
          };
        scanf("%s%d%d%d", stud.rollno, &student.dob.day, &student.dob.month,
        &student.dob.year);
}
```

Answer:

Compiler Error: Undefined structure date

Explanation:

Only declaration of struct date is available inside the structure definition of 'student' but to have a variable of type struct date the definition of the structure is

required.

155) There were 10 records stored in “somefile.dat” but the following program printed 11 names. What went wrong?

```
void main()
{
        struct student
        {
        char name[30], rollno[6];
        }stud;
        FILE *fp = fopen(“somefile.dat”,”r”);
        while(!feof(fp))
         {
       fread(&stud, sizeof(stud), 1 , fp);
                puts(stud.name);
        }
}
```

Explanation:

fread reads 10 records and prints the names successfully. It will return EOF only when fread tries to read another record and fails reading EOF (and returning EOF). So it prints the last record again. After this only the condition feof(fp) becomes false, hence comes out of the while loop.

156) Is there any difference between the two declarations,

1. int foo(int *arr[]) and
2. int foo(int *arr[2])

Answer:

No

Explanation:

Functions can only pass pointers and not arrays. The numbers that are allowed inside the [] is just for more readability. So there is no difference between the two declarations.

157) What is the subtle error in the following code segment?

```
void fun(int n, int arr[])
{
        int *p=0;
        int i=0;
        while(i++<n)
  p = &arr[i];
                *p = 0;
}
```

Answer & Explanation:

If the body of the loop never executes p is assigned no address. So p

remains NULL where *p =0 may result in problem (may rise to runtime error “NULL pointer assignment” and terminate the program).

158) What is wrong with the following code?

```
int *foo()
{
        int *s = malloc(sizeof(int)100);
        assert(s != NULL);
        return s;
}
```

Answer & Explanation:

assert macro should be used for debugging and finding out bugs. The check s != NULL is for error/exception handling and for that assert shouldn’t be used. A plain if and the corresponding remedy statement has to be given.

159) What is the hidden bug with the following statement?

```
assert(val++ != 0);
```

Answer & Explanation:

Assert macro is used for debugging and removed in release version. In assert, the experssion involves side-effects. So the behavior of the code becomes different in case of debug version and the release version thus leading to a subtle bug.

Rule to Remember:

Don’t use expressions that have side-effects in assert statements.

160)

```
void main()
{
int *i = 0x400;   // i points to the address 400
*i = 0;   // set the value of memory location pointed by i;
}
```

Answer:

Undefined behavior

Explanation:

The second statement results in undefined behavior because it points to some location whose value may not be available for modification. *This type of pointer in which the non-availability of the implementation of the referenced location is known as 'incomplete type'.*

161)

```
#define assert(cond) if(!(cond)) \
   (fprintf(stderr, "assertion failed: %s, file %s, line %d \n",#cond,\
  __FILE__,__LINE__), abort())

void main()
{
int i = 10;
if(i==0)
     assert(i < 100);
```

```
else
    printf("This statement becomes else for if in assert macro");
}
```

Answer:

No output

Explanation:

The else part in which the printf is there becomes the else for if in the assert macro. Hence nothing is printed.

The solution is to use conditional operator instead of if statement,

```
#define assert(cond) ((cond)?(0): (fprintf (stderr, "assertion failed: \ %s, file %s, line %d \n",#cond, __FILE__,__LINE__), abort()))
```

Note:

However this problem of "matching with nearest else" cannot be solved by the usual method of placing the if statement inside a block like this,

```
#define assert(cond) { \
if(!(cond)) \
  (fprintf(stderr, "assertion failed: %s, file %s, line %d \n",#cond,\
 __FILE__,__LINE__), abort()) \
}
```

162) Is the following code legal?

```
struct a
    {
        int x;
         struct a b;
    }
```

Answer:

No

Explanation:

Is it not legal for a structure to contain a member that is of the same type as in this case. Because this will cause the structure declaration to be recursive without end.

163) Is the following code legal?

```
struct a
    {
        int x;
            struct a *b;
    }
```

Answer:

Yes.

Explanation:

*b is a pointer to type struct a and so is legal. The compiler knows, the size of the pointer to a structure even before the size of the structure is determined(as you know the pointer to any type is of same size). This type of

structures is known as ‘self-referencing’ structure.

164) Is the following code legal?

```
typedef struct a
    {
        int x;
         aType *b;
    }aType
```

Answer:

No

Explanation:

The typename aType is not known at the point of declaring the structure (forward references are not made for typedefs).

165) Is the following code legal?

```
typedef struct a aType;
struct a
{
        int x;
        aType *b;
};
```

Answer:

Yes

Explanation:

The typename aType is known at the point of declaring the structure, because it is already typedefined.

166) Is the following code legal?

```
void main()
{
        typedef struct a aType;
        aType someVariable;
        struct a
{
        int x;
                aType *b;
       };
}
```

Answer:

No

Explanation:

When the declaration,

```
typedef struct a aType;
```

is encountered body of struct a is not known. This is known as ‘incomplete types’.

167) void main()

```
{
printf("sizeof (void *) = %d \n", sizeof( void *));
printf("sizeof (int *)      = %d \n", sizeof(int *));
printf("sizeof (double *)   = %d \n", sizeof(double *));
printf("sizeof(struct unknown *) = %d \n", sizeof(struct unknown *));
}
```

Answer :

sizeof (void *) = 2
sizeof (int *) = 2
sizeof (double *) = 2
sizeof(struct unknown *) = 2

Explanation:

The pointer to any type is of same size.

168) char inputString[100] = {0};

To get string input from the keyboard which one of the following is better?

1) gets(inputString)
2) fgets(inputString, sizeof(inputString), fp)

Answer & Explanation:

The second one is better because gets(inputString) doesn't know the size of the string passed and so, if a very big input (here, more than 100 chars) the charactes will be written past the input string. When fgets is used with stdin performs the same operation as gets but is safe.

169) Which version do you prefer of the following two,

1) printf("%s",str); // or the more curt one
2) printf(str);

Answer & Explanation:

Prefer the first one. If the str contains any format characters like %d then it will result in a subtle bug.

170)
```
void main()
{
    int i=10, j=2;
    int *ip= &i, *jp = &j;
    int k = *ip/*jp;
    printf("%d",k);
}
```

Answer:

Compiler Error: "Unexpected end of file in comment started in line 5".

Explanation:

The programmer intended to divide two integers, but by the "maximum munch" rule, the compiler treats the operator sequence / and * as /* which happens to be the starting of comment. To force what is intended by the programmer,

```
int k = *ip/ *jp;
```

```
// give space explicity separating / and *
//or
int k = *ip/(*jp);
// put braces to force the intention
```

will solve the problem.

171) void main()

```
{
char ch;
for(ch=0;ch<=127;ch++)
printf("%c    %d \n", ch, ch);
}
```

Answer:

Implementaion dependent

Explanation:

The char type may be signed or unsigned by default. If it is signed then ch++ is executed after ch reaches 127 and rotates back to -128. Thus ch is always smaller than 127.

172) Is this code legal?

```
int *ptr;
ptr = (int *) 0x400;
```

Answer:

Yes

Explanation:

The pointer ptr will point at the integer in the memory location 0x400.

173) main()

```
{
    char a[4]="HELLO";
    printf("%s",a);
}
```

Answer:

Compiler error: Too many initializers

Explanation:

The array a is of size 4 but the string constant requires 6 bytes to get stored.

174) main()

```
{
    char a[4]="HELL";
    printf("%s",a);
}
```

Answer:

HELL%@!~@!@???@~~!

Explanation:

The character array has the memory just enough to hold the string "HELL" and doesnt have enough space to store the terminating null character. So it prints the HELL correctly and continues to print garbage values till it accidentally comes

across a NULL character.

175) main()
{
int a=10,*j;
void *k;
j=k=&a;
j++;
k++;
printf("\n %u %u ",j,k);
}
Answer:
Compiler error: Cannot increment a void pointer
Explanation:
Void pointers are generic pointers and they can be used only when the type is not known and as an intermediate address storage type. No pointer arithmetic can be done on it and you cannot apply indirection operator (*) on void pointers.

176) main()
{
extern int i;
{ int i=20;
{
const volatile unsigned i=30; printf("%d",i);
}
printf("%d",i);
}
printf("%d",i);
}
int i;

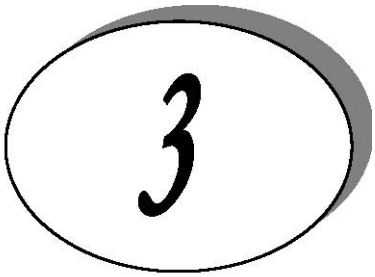
177) Printf can be implemented by using _____ list.
Answer:
Variable length argument lists
178) char *someFun()
{
char *temp = “string constant";
return temp;
}
int main()
{
puts(someFun());
}
Answer:
string constant
Explanation:

The program suffers no problem and gives the output correctly because the character constants are stored in code/data area and not allocated in stack, so this doesn't lead to dangling pointers.

179) char *someFun1()
```
{
char temp[ ] = "string";
return temp;
}
char *someFun2()
{
char temp[ ] = {'s', 't','r','i','n','g'};
return temp;
}
int main()
{
puts(someFun1());
puts(someFun2());
}
```
Answer:
Garbage values.

Explanation:
Both the functions suffer from the problem of dangling pointers. In someFun1() temp is a character array and so the space for it is allocated in heap and is initialized with character string "string". This is created dynamically as the function is called, so is also deleted dynamically on exiting the function so the string data is not available in the calling function main() leading to print some garbage values. The function someFun2() also suffers from the same problem but the problem can be easily identified in this case.

C++ Aptitude and OOPS



Note : All the programs are tested under Turbo C++ 3.0, 4.5 and Microsoft VC++ 6.0

compilers.

It is assumed that,

Programs run under Windows environment,
The underlying machine is an x86 based system,
Program is compiled using Turbo C/C++ compiler.

The program output may depend on the information based on this assumptions (for example sizeof(int) == 2 may be assumed).

1) class Sample

```
{
public:
        int *ptr;
        Sample(int i)
        {
        ptr = new int(i);
        }
        ~Sample()
        {
        delete ptr;
        }
        void PrintVal()
        {
        cout << "The value is " << *ptr;
        }
};
void SomeFunc(Sample x)
{
cout << "Say i am in someFunc " << endl;
}
int main()
{
Sample s1= 10;
SomeFunc(s1);
s1.PrintVal();
}
```

Answer:

Say i am in someFunc
Null pointer assignment(Run-time error)

Explanation:

As the object is passed by value to SomeFunc the destructor of the object is called when the control returns from the function. So when PrintVal is called it meets up with ptr that has been freed.The solution is to pass the Sample object by *reference* to SomeFunc:

```
void SomeFunc(Sample &x)
{
```

```
        cout << "Say i am in someFunc " << endl;
        }
```
because when we pass objects by refernece that object is not destroyed. while returning from the function.

2) Which is the parameter that is added to every non-static member function when it is called?

Answer:
'this' pointer

3)
```
class base
        {
        public:
        int bval;
        base(){ bval=0;}
        };

class deri:public base
        {
        public:
        int dval;
        deri(){ dval=1;}
        };
void SomeFunc(base *arr,int size)
{
for(int i=0; i<size; i++,arr++)
        cout<<arr->bval;
cout<<endl;
}

int main()
{
base BaseArr[5];
SomeFunc(BaseArr,5);
deri DeriArr[5];
SomeFunc(DeriArr,5);
}
```

Answer:
```
        00000
        01010
```
Explanation:

The function SomeFunc expects two arguments.The first one is a pointer to an array of base class objects and the second one is the sizeof the array.The first call of someFunc calls it with an array of bae objects, so it works correctly and prints the bval of all the objects. When Somefunc is called the second time the argument passed is the pointeer to an array of derived

class objects and not the array of base class objects. But that is what the function expects to be sent. So the derived class pointer is promoted to base class pointer and the address is sent to the function. SomeFunc() knows nothing about this and just treats the pointer as an array of base class objects. So when arr++ is met, the size of base class object is taken into consideration and is incremented by sizeof(int) bytes for bval (the deri class objects have bval and dval as members and so is of size >= sizeof(int)+sizeof(int)).

```
4) class base
        {
        public:
          void baseFun(){ cout<<"from base"<<endl;}
        };
 class deri:public base
        {
        public:
          void baseFun(){ cout<< "from derived"<<endl;}
        };
void SomeFunc(base *baseObj)
{
        baseObj->baseFun();
}
int main()
{
base baseObject;
SomeFunc(&baseObject);
deri deriObject;
SomeFunc(&deriObject);
}
```

Answer:

```
 from base
 from base
```

Explanation:

As we have seen in the previous case, SomeFunc expects a pointer to a base class. Since a pointer to a derived class object is passed, it treats the argument only as a base class pointer and the corresponding base function is called.

```
5) class base
        {
        public:
          virtual void baseFun(){ cout<<"from base"<<endl;}
        };
 class deri:public base
        {
        public:
          void baseFun(){ cout<< "from derived"<<endl;}
        };
```

```
void SomeFunc(base *baseObj)
{
        baseObj->baseFun();
}
int main()
{
base baseObject;
SomeFunc(&baseObject);
deri deriObject;
SomeFunc(&deriObject);
}
```

Answer:

from base
from derived

Explanation:

Remember that baseFunc is a virtual function. That means that it supports run-time polymorphism. So the function corresponding to the derived class object is called.

```
void main()
{
 int a, *pa, &ra;
 pa = &a;
 ra = a;
 cout <<"a="<<a <<"*pa="<<*pa <<"ra"<<ra ;
}
/*
Answer :
 Compiler Error: 'ra',reference must be initialized
Explanation :
 Pointers are different from references. One of the main
differences is that the pointers can be both initialized and assigned,
whereas references can only be initialized. So this code issues an error.
*/

const int size = 5;
void print(int *ptr)
{
 cout<<ptr[0];
}

void print(int ptr[size])
{
 cout<<ptr[0];
}
```

```
void main()
{
 int a[size] = {1,2,3,4,5};
 int *b = new int(size);
 print(a);
 print(b);
}
/*
Answer:
 Compiler Error : function 'void print(int *)' already has a body

Explanation:
 Arrays cannot be passed to functions, only pointers (for arrays, base addresses)
can be passed. So the arguments int *ptr and int prt[size] have no difference
as function arguments. In other words, both the functoins have the same signature and
so cannot be overloaded.
*/

class some{
public:
 ~some()
 {
  cout<<"some's destructor"<<endl;
 }
};

void main()
{
 some s;
 s.~some();
}
/*
Answer:
 some's destructor
 some's destructor
Explanation:
 Destructors can be called explicitly. Here 's.~some()' explicitly calls the
destructor of 's'. When main() returns, destructor of s is called again,
hence the result.
*/

#include <iostream.h>

class fig2d
{
 int dim1;
```

```
 int dim2;

public:
 fig2d() { dim1=5; dim2=6;}

 virtual void operator<<(ostream & rhs);
};

void fig2d::operator<<(ostream &rhs)
{
 rhs <<this->dim1<<" "<<this->dim2<<" ";
}

/*class fig3d : public fig2d
{
 int dim3;
public:
 fig3d() { dim3=7;}
 virtual void operator<<(ostream &rhs);
};
void fig3d::operator<<(ostream &rhs)
{
 fig2d::operator <<(rhs);
 rhs<<this->dim3;
}
*/

void main()
{
 fig2d obj1;
// fig3d obj2;

 obj1 << cout;
// obj2 << cout;
}
/*
Answer :
 5 6
Explanation:
 In this program, the << operator is overloaded with ostream as argument.
This enables the 'cout' to be present at the right-hand-side. Normally, 'cout'
is implemented as global function, but it doesn't mean that 'cout' is not possible
to be overloaded as member function.
     Overloading << as virtual member function becomes handy when the class in which
it is overloaded is inherited, and this becomes available to be overrided. This is as opposed
to global friend functions, where friend's are not inherited.
```

```
*/

class opOverload{
public:
 bool operator==(opOverload temp);
};

bool opOverload::operator==(opOverload temp){
 if(*this   == temp ){
  cout<<"The both are same objects\n";
  return true;
 }
 else{
  cout<<"The both are different\n";
  return false;
 }
}

void main(){
 opOverload a1, a2;
 a1= =a2;
}
```

Answer :
 Runtime Error: Stack Overflow
Explanation :
 Just like normal functions, operator functions can be called recursively. This program just illustrates that point, by calling the operator == function recursively, leading to an infinite loop.

```
class complex{
 double re;
 double im;
public:
 complex() : re(1),im(0.5) {}
 bool operator==(complex &rhs);
 operator int(){}
};

bool complex::operator == (complex &rhs){
 if((this->re == rhs.re) && (this->im == rhs.im))
  return true;
 else
  return false;
}
```

```
int main(){
 complex   c1;
 cout<<   c1;
}
```

Answer : Garbage value

Explanation:
The programmer wishes to print the complex object using output re-direction operator,which he has not defined for his lass.But the compiler instead of giving an error sees the conversion function and converts the user defined object to standard object and prints some garbage value.

```
class complex{
 double re;
 double im;
public:
 complex() : re(0),im(0) {}
 complex(double n) { re=n,im=n;};
 complex(int m,int n) { re=m,im=n;}
 void print() { cout<<re; cout<<im;}
};

void main(){
 complex c3;
 double i=5;
 c3 = i;
 c3.print();
}
```

Answer:
5,5
Explanation:
Though no operator= function taking complex, double is defined, the double on the rhs is converted into a temporary object using the single argument constructor taking double and assigned to the lvalue.

```
void main()
{
 int a, *pa, &ra;
 pa = &a;
 ra = a;
 cout <<"a="<<a <<"*pa="<<*pa <<"ra"<<ra ;
```

```
}
```

Answer :
Compiler Error: 'ra',reference must be initialized
Explanation :
Pointers are different from references. One of the main differences is that the pointers can be both initialized and assigned, whereas references can only be initialized. So this code issues an error.

Try it Yourself

1) Determine the output of the 'C++' Codelet.

```
class base
{
public :
  out()
  {
    cout<<"base ";
  }
};
class deri{
public : out()
{
cout<<"deri ";
}
};
void main()
{    deri dp[3];
  base *bp = (base*)dp;
  for (int i=0; i<3;i++)
  (bp++)->out();
}
```

2) Justify the use of virtual constructors and destructors in C++.

3) Each C++ object possesses the 4 member fns,(which can be declared by the programmer explicitly or by the implementation if they are not available). What are those 4 functions?

4) What is wrong with this class declaration?

```
class something
{
  char *str;
  public:
       something(){
       st = new char[10]; }
```

```
    ~something()
    {
      delete str;
    }
  };
```

5) Inheritance is also known as -------- relationship. Containership as _____ relationship.

6) When is it necessary to use member-wise initialization list (also known as header initialization list) in C++?

7) Which is the only operator in C++ which can be overloaded but NOT inherited.

8) Is there anything wrong with this C++ class declaration?

```
 class temp
 {
   int value1;
   mutable int value2;
   public :
    void fun(int val)
  const{
    ((temp*) this)->value1 = 10;
    value2 = 10;
    }
  };
```

1. What is a modifier?

Answer:

A modifier, also called a modifying function is a member function that changes the value of at least one data member. In other words, an operation that modifies the state of an object. Modifiers are also known as 'mutators'.

2. What is an accessor?

Answer:

An accessor is a class operation that does not modify the state of an object. The accessor functions need to be declared as *const* operations

3. Differentiate between a template class and class template.

Answer:

Template class:

A generic definition or a parameterized class not instantiated until the client provides the needed information. It's jargon for plain templates.

Class template:

A class template specifies how individual classes can be constructed much like the way a class specifies how individual objects can be constructed. It's jargon for plain classes.

4. When does a name clash occur?

Answer:

A name clash occurs when a name is defined in more than one place. For example., two different class libraries could give two different classes the same name. If you try to use many class libraries at the same time, there is a fair chance that you will be unable to compile or link the program because of name clashes.

5. Define namespace.

Answer:

It is a feature in c++ to minimize name collisions in the global name space. This namespace keyword assigns a distinct name to a library that allows other libraries to use the same identifier names without creating any name collisions. Furthermore, the compiler uses the namespace signature for differentiating the definitions.

6. What is the use of 'using' declaration.

Answer:

A using declaration makes it possible to use a name from a namespace without the scope operator.

7. What is an Iterator class?

Answer:

A class that is used to traverse through the objects maintained by a container class. There are five categories of iterators:

input iterators,
output iterators,
forward iterators,
bidirectional iterators,
random access.

An iterator is an entity that gives access to the contents of a container object without violating encapsulation constraints. Access to the contents is granted on a one-at-a-time basis in order. The order can be storage order (as in lists and queues) or some arbitrary order (as in array indices) or according to some ordering relation (as in an ordered binary tree). The iterator is a construct, which provides an interface that, when called, yields either the next element in the container, or some value denoting the fact that there are no more elements to examine. Iterators hide the details of access to and update of the elements of a container class.

The simplest and safest iterators are those that permit read-only access to the contents of a container class. The following code fragment shows how an iterator might appear in code:

```
cont_iter:=new cont_iterator();
x:=cont_iter.next();
while x/=none do
        ...
        s(x);
        ...
        x:=cont_iter.next();
end;
```

In this example, cont_iter is the name of the iterator. It is created on the first line by instantiation of cont_iterator class, an iterator class defined to iterate over some container class,

cont. Succesive elements from the container are carried to x. The loop terminates when x is bound to some empty value. (Here, none)In the middle of the loop, there is s(x) an operation on x, the current element from the container. The next element of the container is obtained at the bottom of the loop.

9. List out some of the OODBMS available.

Answer:

GEMSTONE/OPAL of Gemstone systems.
ONTOS of Ontos.
Objectivity of Objectivity inc.
Versant of Versant object technology.
Object store of Object Design.
ARDENT of ARDENT software.
POET of POET software.

10. List out some of the object-oriented methodologies.

Answer:

Object Oriented Development (OOD) (Booch 1991,1994).
Object Oriented Analysis and Design (OOA/D) (Coad and Yourdon 1991).
Object Modelling Techniques (OMT) (Rumbaugh 1991).
Object Oriented Software Engineering (Objectory) (Jacobson 1992).
Object Oriented Analysis (OOA) (Shlaer and Mellor 1992).
The Fusion Method (Coleman 1991).

11. What is an incomplete type?

Answer:

Incomplete types refers to pointers in which there is non availability of the implementation of the referenced location or it points to some location whose value is not available for modification.

Example:

```
int *i=0x400   // i points to address 400
*i=0;          //set the value of memory location pointed by i.
```

Incomplete types are otherwise called uninitialized pointers.

12. What is a dangling pointer?

Answer:

A dangling pointer arises when you use the address of an object after its lifetime is over. This may occur in situations like returning addresses of the automatic variables from a function or using the address of the memory block after it is freed.

13. Differentiate between the message and method.

Answer:

Message	***Method***
Objects communicate by sending messages	Provides response to a message.

to each other.
A message is sent to invoke a method. It is an implementation of an operation.

14. What is an adaptor class or Wrapper class?

Answer:

A class that has no functionality of its own. Its member functions hide the use of a third party software component or an object with the non-compatible interface or a non- object-oriented implementation.

15. What is a Null object?

Answer:

It is an object of some class whose purpose is to indicate that a real object of that class does not exist. One common use for a null object is a return value from a member function that is supposed to return an object with some specified properties but cannot find such an object.

16. What is class invariant?

Answer:

A class invariant is a condition that defines all valid states for an object. It is a logical condition to ensure the correct working of a class. Class invariants must hold when an object is created, and they must be preserved under all operations of the class. In particular all class invariants are both preconditions and post-conditions for all operations or member functions of the class.

17. What do you mean by Stack unwinding?

Answer:

It is a process during exception handling when the destructor is called for all local objects between the place where the exception was thrown and where it is caught.

18. Define precondition and post-condition to a member function.

Answer:

Precondition:

A precondition is a condition that must be true on entry to a member function. A class is used correctly if preconditions are never false. An operation is not responsible for doing anything sensible if its precondition fails to hold.

For example, the interface invariants of *stack class* say nothing about pushing yet another element on a stack that is already full. We say that *isful()* is a precondition of the *push* operation.

Post-condition:

A post-condition is a condition that must be true on exit from a member function if the precondition was valid on entry to that function. A class is implemented correctly if post-conditions are never false.

For example, after pushing an element on the stack, we know that *isempty()* must necessarily hold. This is a post-condition of the *push* operation.

19. What are the conditions that have to be met for a condition to be an invariant of the class?

Answer:

The condition should hold at the end of every constructor.

The condition should hold at the end of every mutator(non-const) operation.

20. What are proxy objects?

Answer:

Objects that stand for other objects are called proxy objects or surrogates.

Example:

```
template<class T>
class Array2D
{
        public:
            class Array1D
             {
public:
                T& operator[] (int index);
                const T& operator[] (int index) const;
                ...
             };
            Array1D operator[] (int index);
            const Array1D operator[] (int index) const;
            ...
 };
```

The following then becomes legal:

```
    Array2D<float>data(10,20);
.....
cout<<data[3][6];       //   fine
```

Here data[3] yields an Array1D object and the operator [] invocation on that object yields the float in position(3,6) of the original two dimensional array. Clients of the Array2D class need not be aware of the presence of the Array1D class. Objects of this latter class stand for one-dimensional array objects that, conceptually, do not exist for clients of Array2D. Such clients program as if they were using real, live, two-dimensional arrays. Each Array1D object stands for a one-dimensional array that is absent from a conceptual model used by the clients of Array2D. In the above example, Array1D is a proxy class. Its instances stand for one-dimensional arrays that, conceptually, do not exist.

21. Name some pure object oriented languages.

Answer:

Smalltalk,
Java,
Eiffel,
Sather.

22. Name the operators that cannot be overloaded.

Answer:

sizeof . .* .-> :: ?:

23. What is a node class?

Answer:

A node class is a class that,
relies on the base class for services and implementation,
provides a wider interface to te users than its base class,
relies primarily on virtual functions in its public interface
depends on all its direct and indirect base class
can be understood only in the context of the base class
can be used as base for further derivation
can be used to create objects.

A node class is a class that has added new services or functionality beyond the services inherited from its base class.

24. What is an orthogonal base class?

Answer:

If two base classes have no overlapping methods or data they are said to be independent of, or orthogonal to each other. Orthogonal in the sense means that two classes operate in different dimensions and do not interfere with each other in any way. The same derived class may inherit such classes with no difficulty.

25. What is a container class? What are the types of container classes?

Answer:

A container class is a class that is used to hold objects in memory or external storage. A container class acts as a generic holder. A container class has a predefined behavior and a well-known interface. A container class is a supporting class whose purpose is to hide the topology used for maintaining the list of objects in memory. When a container class contains a group of mixed objects, the container is called a heterogeneous container; when the container is holding a group of objects that are all the same, the container is called a homogeneous container.

26. What is a protocol class?

Answer:

An abstract class is a protocol class if:

it neither contains nor inherits from classes that contain member data, non-virtual functions, or private (or protected) members of any kind.

it has a non-inline virtual destructor defined with an empty implementation,

all member functions other than the destructor including inherited functions, are declared pure virtual functions and left undefined.

27. What is a mixin class?

Answer:

A class that provides some but not all of the implementation for a virtual base class is

often called mixin. Derivation done just for the purpose of redefining the virtual functions in the base classes is often called mixin inheritance. Mixin classes typically don't share common bases.

28. What is a concrete class?

Answer:

A concrete class is used to define a useful object that can be instantiated as an automatic variable on the program stack. The implementation of a concrete class is defined. The concrete class is not intended to be a base class and no attempt to minimize dependency on other classes in the implementation or behavior of the class.

29.What is the handle class?

Answer:

A handle is a class that maintains a pointer to an object that is programmatically accessible through the public interface of the handle class.

Explanation:

In case of abstract classes, unless one manipulates the objects of these classes through pointers and references, the benefits of the virtual functions are lost. User code may become dependent on details of implementation classes because an abstract type cannot be allocated statistically or on the stack without its size being known. Using pointers or references implies that the burden of memory management falls on the user. Another limitation of abstract class object is of fixed size. Classes however are used to represent concepts that require varying amounts of storage to implement them.

A popular technique for dealing with these issues is to separate what is used as a single object in two parts: a handle providing the user interface and a representation holding all or most of the object's state. The connection between the handle and the representation is typically a pointer in the handle. Often, handles have a bit more data than the simple representation pointer, but not much more. Hence the layout of the handle is typically stable, even when the representation changes and also that handles are small enough to move around relatively freely so that the user needn’t use the pointers and the references.

30. What is an action class?

Answer:

The simplest and most obvious way to specify an action in C++ is to write a function. However, if the action has to be delayed, has to be transmitted 'elsewhere' before being performed, requires its own data, has to be combined with other actions, etc then it often becomes attractive to provide the action in the form of a class that can execute the desired action and provide other services as well. Manipulators used with iostreams is an obvious example.

Explanation:

A common form of action class is a simple class containing just one virtual function.

```
class Action
{
        public:
            virtual int do_it( int )=0;
            virtual ~Action( );
  }
```

Given this, we can write code say a member that can store actions for later execution

without using pointers to functions, without knowing anything about the objects involved, and without even knowing the name of the operation it invokes. For example:

```
class write_file : public Action
    {
            File& f;
            public:
                int do_it(int)
                {
                        return fwrite( ).suceed( );
                }
     };
    class error_message: public Action
    {
                response_box db(message.cstr( ),"Continue","Cancel","Retry");
                switch (db.getresponse( ))
                {
                        case 0: return 0;
                        case 1: abort();
                        case 2: current_operation.redo( );return 1;
                 }
     };
```

A user of the Action class will be completely isolated from any knowledge of derived classes such as write_file and error_message.

31. When can you tell that a memory leak will occur?

Answer:

A memory leak occurs when a program loses the ability to free a block of dynamically allocated memory.

32.What is a parameterized type?

Answer:

A template is a parameterized construct or type containing generic code that can use or manipulate any type. It is called parameterized because an actual type is a parameter of the code body. Polymorphism may be achieved through parameterized types. This type of polymorphism is called parameteric polymorphism. Parameteric polymorphism is the mechanism by which the same code is used on different types passed as parameters.

33. Differentiate between a deep copy and a shallow copy?

Answer:

Deep copy involves using the contents of one object to create another instance of the same class. In a deep copy, the two objects may contain ht same information but the target object will have its own buffers and resources. the destruction of either object will not affect the remaining object. The overloaded assignment operator would create a deep copy of objects.

Shallow copy involves copying the contents of one object into another instance of the same class thus creating a mirror image. Owing to straight copying of references and pointers,

the two objects will share the same externally contained contents of the other object to be unpredictable.

Explanation:

Using a copy constructor we simply copy the data values member by member. This method of copying is called shallow copy. If the object is a simple class, comprised of built in types and no pointers this would be acceptable. This function would use the values and the objects and its behavior would not be altered with a shallow copy, only the addresses of pointers that are members are copied and not the value the address is pointing to. The data values of the object would then be inadvertently altered by the function. When the function goes out of scope, the copy of the object with all its data is popped off the stack.

If the object has any pointers a deep copy needs to be executed. With the deep copy of an object, memory is allocated for the object in free store and the elements pointed to are copied. A deep copy is used for objects that are returned from a function.

34. What is an opaque pointer?

Answer:

A pointer is said to be opaque if the definition of the type to which it points to is not included in the current translation unit. A translation unit is the result of merging an implementation file with all its headers and header files.

35. What is a smart pointer?

Answer:

A smart pointer is an object that acts, looks and feels like a normal pointer but offers more functionality. In C++, smart pointers are implemented as *template classes* that encapsulate a pointer and override standard pointer operators. They have a number of advantages over regular pointers. They are guaranteed to be initialized as either null pointers or pointers to a heap object. Indirection through a null pointer is checked. No delete is ever necessary. Objects are automatically freed when the last pointer to them has gone away. One significant problem with these smart pointers is that unlike regular pointers, they don't respect inheritance. Smart pointers are unattractive for polymorphic code. Given below is an example for the implementation of smart pointers.

Example:

```
template <class X>
class smart_pointer
{
        public:
            smart_pointer();                        // makes a null pointer
            smart_pointer(const X& x)               // makes pointer to copy of x

            X& operator *( );
            const X& operator*( ) const;
            X* operator->() const;

            smart_pointer(const smart_pointer <X> &);
            const smart_pointer <X> & operator =(const smart_pointer<X>&);
            ~smart_pointer();
        private:
```

```
        //...
};
```

This class implement a smart pointer to an object of type X. The object itself is located on the heap. Here is how to use it:

```
smart_pointer <employee> p= employee("Harris",1333);
```

Like other overloaded operators, p will behave like a regular pointer,

```
cout<<*p;
p->raise_salary(0.5);
```

36. What is reflexive association?

Answer:

The 'is-a' is called a reflexive association because the reflexive association permits classes to bear the is-a association not only with their super-classes but also with themselves. It differs from a 'specializes-from' as 'specializes-from' is usually used to describe the association between a super-class and a sub-class. For example:

Printer is-a printer.

37. What is slicing?

Answer:

Slicing means that the data added by a subclass are discarded when an object of the subclass is passed or returned by value or from a function expecting a base class object.

Explanation:

Consider the following class declaration:

```
class base
{
        ...
        base& operator =(const base&);
        base (const base&);
}
void fun( )
{
      base e=m;
      e=m;
}
```

As base copy functions don't know anything about the derived only the base part of the derived is copied. This is commonly referred to as slicing. One reason to pass objects of classes in a hierarchy is to avoid slicing. Other reasons are to preserve polymorphic behavior and to gain efficiency.

38. What is name mangling?

Answer:

Name mangling is the process through which your c++ compilers give each function in your program a unique name. In C++, all programs have at-least a few functions with the same name. Name mangling is a concession to the fact that linker always insists on all function names being unique.

Example:

In general, member names are made unique by concatenating the name of the member with that

of the class e.g. given the declaration:

```
class Bar
 {
        public:
            int ival;
            ...
  };
```

ival becomes something like:

```
  // a possible member name mangling
 ival__3Bar
```

Consider this derivation:

```
 class Foo : public Bar
{
      public:
          int ival;
          ...
}
```

The internal representation of a Foo object is the concatenation of its base and derived class members.

```
 // Pseudo C++ code
// Internal representation of Foo
class Foo
{
     public:
         int ival__3Bar;
         int ival__3Foo;
         ...
};
```

Unambiguous access of either ival members is achieved through name mangling. Member functions, because they can be overloaded, require an extensive mangling to provide each with a unique name. Here the compiler generates the same name for the two overloaded instances(Their argument lists make their instances unique).

39. What are proxy objects?

Answer:

Objects that points to other objects are called proxy objects or surrogates. Its an object that provides the same interface as its server object but does not have any functionality. During a method invocation, it routes data to the true server object and sends back the return value to the object.

40. Differentiate between declaration and definition in C++.

Answer:

A declaration introduces a name into the program; a definition provides a unique description of an entity (e.g. type, instance, and function). Declarations can be repeated in a given scope, it introduces a name in a given scope. There must be exactly one definition of every object, function or class used in a C++ program.

A declaration is a definition unless:

it declares a function without specifying its body,
it contains an extern specifier and no initializer or function body,
it is the declaration of a static class data member without a class definition,
it is a class name definition,
it is a typedef declaration.

A definition is a declaration unless:
it defines a static class data member,
it defines a non-inline member function.

41. What is cloning?

Answer:

An object can carry out copying in two ways i.e. it can set itself to be a copy of another object, or it can return a copy of itself. The latter process is called cloning.

42. Describe the main characteristics of static functions.

Answer:

The main characteristics of static functions include,
It is without the a this pointer,
It can't directly access the non-static members of its class
It can't be declared const, volatile or virtual.
It doesn't need to be invoked through an object of its class, although for convenience, it may.

43. Will the inline function be compiled as the inline function always? Justify.

Answer:

An inline function is a request and not a command. Hence it won't be compiled as an inline function always.

Explanation:

Inline-expansion could fail if the inline function contains loops, the address of an inline function is used, or an inline function is called in a complex expression. The rules for inlining are compiler dependent.

44. Define a way other than using the keyword inline to make a function inline.

Answer:

The function must be defined inside the class.

45. How can a '::' operator be used as unary operator?

Answer:

The scope operator can be used to refer to members of the global namespace. Because the global namespace doesn't have a name, the notation :: member-name refers to a member of the global namespace. This can be useful for referring to members of global namespace whose names have been hidden by names declared in nested local scope. Unless we specify to the compiler in which namespace to search for a declaration, the compiler simple searches the

current scope, and any scopes in which the current scope is nested, to find the declaration for the name.

46. What is placement new?

Answer:

When you want to call a constructor directly, you use the placement new. Sometimes you have some raw memory that's already been allocated, and you need to construct an object in the memory you have. Operator new's special version placement new allows you to do it.

```
class Widget
{
    public :
        Widget(int widgetsize);
         ...
         Widget* Construct_widget_int_buffer(void *buffer,int widgetsize)
          {
              return new(buffer) Widget(widgetsize);
          }
};
```

This function returns a pointer to a Widget object that's constructed within the buffer passed to the function. Such a function might be useful for applications using shared memory or memory-mapped I/O, because objects in such applications must be placed at specific addresses or in memory allocated by special routines.

OOAD

1. What do you mean by analysis and design?

Analysis:

Basically, it is the process of determining what needs to be done before how it should be done. In order to accomplish this, the developer refers the existing systems and documents. So, simply it is an art of discovery.

Design:

It is the process of adopting/choosing the one among the many, which best accomplishes the users needs. So, simply, it is compromising mechanism.

2. What are the steps involved in designing?

Before getting into the design the designer should go through the SRS prepared by the System Analyst.

The main tasks of design are Architectural Design and Detailed Design.

In Architectural Design we find what are the main modules in the problem domain.

In Detailed Design we find what should be done within each module.

3. What are the main underlying concepts of object orientation?

Objects, messages, class, inheritance and polymorphism are the main concepts of object

orientation.

4. What do u meant by "SBI" of an object?

SBI stands for State, Behavior and Identity. Since every object has the above three.

State:

It is just a value to the attribute of an object at a particular time.

Behaviour:

It describes the actions and their reactions of that object.

Identity:

An object has an identity that characterizes its own existence. The identity makes it possible to distinguish any object in an unambiguous way, and independently from its state.

5. Differentiate persistent & non-persistent objects?

Persistent refers to an object's ability to transcend time or space. A persistent object stores/saves its state in a permanent storage system with out losing the information represented by the object.

A non-persistent object is said to be transient or ephemeral. By default objects are considered as non-persistent.

6. What do you meant by active and passive objects?

Active objects are one which instigate an interaction which owns a thread and they are responsible for handling control to other objects. In simple words it can be referred as *client*.

Passive objects are one, which passively waits for the message to be processed. It waits for another object that requires its services. In simple words it can be referred as *server*.

Diagram:

client server
(Active) (Passive)

7. What is meant by software development method?

Software development method describes how to model and build software systems in a reliable and reproducible way. To put it simple, methods that are used to represent ones' thinking using graphical notations.

8. What are models and meta models?

Model:

It is a complete description of something (i.e. system).

Meta model:

It describes the model elements, syntax and semantics of the notation that allows their manipulation.

9. What do you meant by static and dynamic modeling?

Static modeling is used to specify structure of the objects that exist in the problem domain. These are expressed using *class*, *object* and *USECASE diagrams*.

But Dynamic modeling refers representing the object interactions during runtime. It is represented by *sequence*, *activity*, *collaboration* and *statechart diagrams*.

10. How to represent the interaction between the modeling elements?

Model element is just a notation to represent (Graphically) the entities that exist in the problem domain. e.g. for modeling element is class notation, object notation etc.

Relationships are used to represent the interaction between the modeling elements.

The following are the Relationships.

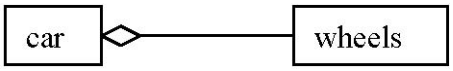
Association: Its' just a semantic connection two classes.

e.g.:

Aggregation: Its' the relationship between two classes which are related in the fashion that *master and slave*. The master takes full rights than the slave. Since the slave works under the master. It is represented as line with diamond in the master area.

ex:

car contains wheels, etc.



car

Containment: This relationship is applied when the part contained with in the whole part, dies when the whole part dies.

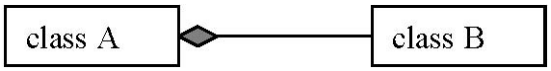
It is represented as darked diamond at the whole part.

example:

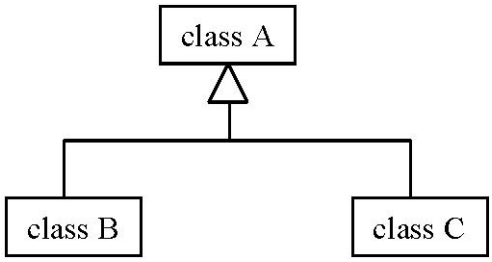
```
class A{
            //some code
};

class B
            {
          A aa; // an object of class A;
         // some code for class B;
};
```

In the above example we see that an object of class A is instantiated with in the class B. so the object class A dies when the object class B dies.we can represnt it in diagram like this.



Generalization: This relationship used when we want represents a class, which captures the common states of objects of different classes. It is represented as arrow line pointed at the class, which has captured the common states.



Dependency: It is the relationship between dependent and independent classes. Any change in the independent class will affect the states of the dependent class.

DIAGRAM:

class A class B

11. Why generalization is very strong?

Even though Generalization satisfies Structural, Interface, Behaviour properties. It is mathematically very strong, as it is Antisymmetric and Transitive.

Antisymmetric: employee is a person, but not all persons are employees. Mathematically all As' are B, but all Bs' not A.

Transitive: A=>B, B=>c then A=>c.

A. Salesman.

B. Employee.

C. Person.

Note: All the other relationships satisfy all the properties like Structural properties, Interface properties, Behaviour properties.

12. Differentiate Aggregation and containment?

Aggregation is the relationship between the whole and a part. We can add/subtract some properties in the part (slave) side. It won't affect the whole part.

Best example is Car, which contains the wheels and some extra parts. Even though the parts are not there we can call it as car.

But, in the case of containment the whole part is affected when the part within that got affected. The human body is an apt example for this relationship. When the whole body dies the parts (heart etc) are died.

13. Can link and Association applied interchangeably?

No, You cannot apply the link and Association interchangeably. Since link is used represent the relationship between the two objects.

But Association is used represent the relationship between the two classes.

link :: student:Abhilash course:MCA

Association:: student course

14. what is meant by "method-wars"?

Before 1994 there were different methodologies like Rumbaugh, Booch, Jacobson, Meyer etc who followed their own notations to model the systems. The developers were in a dilemma to choose the method which best accomplishes their needs. This particular span was called as "method-wars"

15. Whether unified method and unified modeling language are same or different?

Unified method is convergence of the Rumbaugh and Booch.

Unified modeling lang. is the fusion of Rumbaugh, Booch and Jacobson as well as Betrand Meyer (whose contribution is "sequence diagram"). Its' the superset of all the methodologies.

16. Who were the three famous amigos and what was their contribution to the object community?

The Three amigos namely,

James Rumbaugh (OMT): A veteran in analysis who came up with an idea about the objects and their Relationships (in particular Associations).

Grady Booch: A veteran in design who came up with an idea about partitioning of systems into subsystems.

Ivar Jacobson (Objectory): The father of USECASES, who described about the user and system interaction.

17. Differentiate the class representation of Booch, Rumbaugh and UML?

If you look at the class representaiton of Rumbaugh and UML, It is some what similar and both are very easy to draw.

Representation: OMT UML.

Diagram:

Booch: In this method classes are represented as "Clouds" which are not very easy to draw as for as the developer's view is concern.

Diagram:

18. What is an USECASE? Why it is needed?

A Use Case is a description of a set of sequence of actions that a system performs that yields an observable result of value to a particular action.
In SSAD process <=> In OOAD USECASE. It is represented elliptically.

Representation:

19. Who is an Actor?

An Actor is someone or something that must interact with the system.In addition to that an Actor initiates the process(that is USECASE).

It is represented as a stickman like this.

Diagram:

20. What is guard condition?

Guard condition is one, which acts as a firewall. The access from a particular object can be made only when the particular condition is met.

For Example,

customer check customer number ATM.

Here the object on the customer accesses the ATM facility only when the guard condition is met.

21. Differentiate the following notations?

I: :obj1 :obj2

II: :obj1 :obj2

In the above representation I, obj1 sends message to obj2. But in the case of II the data is transferred from obj1 to obj2.

22. USECASE is an implementation independent notation. How will the designer give the implementation details of a particular USECASE to the programmer?

This can be accomplished by specifying the relationship called "refinement" which talks about the two different abstraction of the same thing.

Or example,

calculate pay calculate

class1 class2 class3

23. Suppose a class acts an Actor in the problem domain, how to represent it in the static model?

In this scenario you can use "stereotype". Since stereotype is just a string that gives extra semantic to the particular entity/model element. It is given with in the << >>.

```
class A
<< Actor>>
attributes

methods.
```

24. Why does the function arguments are called as "signatures"?

The arguments distinguish functions with the same name (functional polymorphism). The name alone does not necessarily identify a unique function. However, the name and its arguments (signatures) will uniquely identify a function.

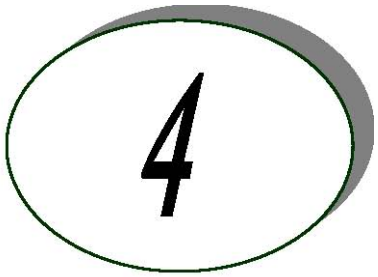
In real life we see suppose, in class there are two guys with same name, but they can be easily identified by their signatures. The same concept is applied here.

ex:

```
class person
{
        public:
 char getsex();
 void setsex(char);
 void setsex(int);
};
```

In the above example we see that there is a function setsex() with same name but with different signature.

Quantitative Aptitude



Exercise 1
Solve the following and check with the answers given at the end.

1. It was calculated that 75 men could complete a piece of work in 20 days. When work was scheduled to commence, it was found necessary to send 25 men to another project. How much longer will it take to complete the work?

2. A student divided a number by 2/3 when he required to multiply by 3/2. Calculate the percentage of error in his result.

3. A dishonest shopkeeper professes to sell pulses at the cost price, but he uses a false weight of 950gm. for a kg. His gain is …%.

4. A software engineer has the capability of thinking 100 lines of code in five minutes and can type 100 lines of code in 10 minutes. He takes a break for five minutes after every ten minutes. How many lines of codes will he complete typing after an hour?

5. A man was engaged on a job for 30 days on the condition that he would get a wage of Rs. 10 for the day he works, but he have to pay a fine of Rs. 2 for each day of his absence. If he gets Rs. 216 at the end, he was absent for work for ... days.

6. A contractor agreeing to finish a work in 150 days, employed 75 men each working 8 hours

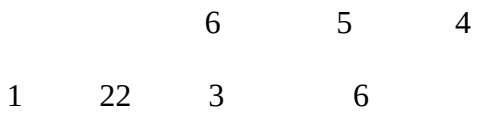
daily. After 90 days, only 2/7 of the work was completed. Increasing the number of men by _____ each working now for 10 hours daily, the work can be completed in time.

7. what is a percent of b divided by b percent of a?
(a) a (b) b (c) 1 (d) 10 (d) 100

8. A man bought a horse and a cart. If he sold the horse at 10 % loss and the cart at 20 % gain, he would not lose anything; but if he sold the horse at 5% loss and the cart at 5% gain, he would lose Rs. 10 in the bargain. The amount paid by him was Rs._____ for the horse and Rs._____ for the cart.

9. A tennis marker is trying to put together a team of four players for a tennis tournament out of seven available. males - a, b and c; females – m, n, o and p. All players are of equal ability and there must be at least two males in the team. For a team of four, all players must be able to play with each other under the following restrictions:
b should not play with m,
c should not play with p, and
a should not play with o.
Which of the following statements must be false?
1. b and p cannot be selected together
2. c and o cannot be selected together
3. c and n cannot be selected together.

10-12. The following figure depicts three views of a cube. Based on this, answer questions 10-12.



10. The number on the face opposite to the face carrying 1 is _____ .

11. The number on the faces adjacent to the face marked 5 are _____ .

12. Which of the following pairs does not correctly give the numbers on the opposite faces.
(1) 6,5 (2) 4,1 (3) 1,3 (4) 4,2

13. Five farmers have 7, 9, 11, 13 & 14 apple trees, respectively in their orchards. Last year, each of them discovered that every tree in their own orchard bore exactly the same number of apples. Further, if the third farmer gives one apple to the first, and the fifth gives three to each of the second and the fourth, they would all have exactly the same number of apples. What were the yields per tree in the orchards of the third and fourth farmers?

14. Five boys were climbing a hill. J was following H. R was just ahead of G. K was between G & H. They were climbing up in a column. Who was the second?

15-18 John is undecided which of the four novels to buy. He is considering a spy thriller, a Murder mystery, a Gothic romance and a science fiction novel. The books are written by Rothko, Gorky, Burchfield and Hopper, not necessary in that order, and published by Heron, Piegon, Blueja and sparrow, not necessary in that order.

(1) The book by Rothko is published by Sparrow.
(2) The Spy thriller is published by Heron.
(3) The science fiction novel is by Burchfield and is not published by Blueja.
(4)The Gothic romance is by Hopper.

15. Pigeon publishes _____.

16. The novel by Gorky _____.

17. John purchases books by the authors whose names come first and third in alphabetical order. He does not buy the books _____.

18. On the basis of the first paragraph and statement (2), (3) and (4) only, it is possible to deduce that

1. Rothko wrote the murder mystery or the spy thriller
2. Sparrow published the murder mystery or the spy thriller
3. The book by Burchfield is published by Sparrow.

19. If a light flashes every 6 seconds, how many times will it flash in ¾ of an hour?

20. If point P is on line segment AB, then which of the following is always true?
(1) AP = PB (2) AP > PB (3) PB > AP (4) AB > AP (5) AB > AP + PB

21. All men are vertebrates. Some mammals are vertebrates. Which of the following conclusions drawn from the above statement is correct.

All men are mammals
All mammals are men
Some vertebrates are mammals.
None

22. Which of the following statements drawn from the given statements are correct?
Given:

All watches sold in that shop are of high standard. Some of the HMT watches are sold in that shop.

a) All watches of high standard were manufactured by HMT.
b) Some of the HMT watches are of high standard.
c) None of the HMT watches is of high standard.

d) Some of the HMT watches of high standard are sold in that shop.

23-27.

1. Ashland is north of East Liverpool and west of Coshocton.
2. Bowling green is north of Ashland and west of Fredericktown.
3. Dover is south and east of Ashland.
4. East Liverpool is north of Fredericktown and east of Dover.
5. Fredericktown is north of Dover and west of Ashland.
6. Coshocton is south of Fredericktown and west of Dover.

23. Which of the towns mentioned is furthest of the north – west
(a) Ashland (b) Bowling green (c) Coshocton
(d) East Liverpool (e) Fredericktown

24. Which of the following must be both north and east of Fredericktown?
(a) Ashland (b) Coshocton (c) East Liverpool
I a only II b only III c only IV a & b V a & c

25. Which of the following towns must be situated both south and west of at least one other town?

A. Ashland only
B. Ashland and Fredericktown
C. Dover and Fredericktown
D. Dover, Coshocton and Fredericktown
E. Coshocton, Dover and East Liverpool.

26. Which of the following statements, if true, would make the information in the numbered statements more specific?

(a) Coshocton is north of Dover.
(b) East Liverpool is north of Dover
(c) Ashland is east of Bowling green.
(d) Coshocton is east of Fredericktown
(e) Bowling green is north of Fredericktown

27. Which of the numbered statements gives information that can be deduced from one or more of the other statements?
(A) 1 (B) 2 (C) 3 (D) 4 (E) 6

28. Eight friends Harsha, Fakis, Balaji, Eswar, Dhinesh, Chandra, Geetha, and Ahmed are sitting in a circle facing the center. Balaji is sitting between Geetha and Dhinesh. Harsha is third to the left of Balaji and second to the right of Ahmed. Chandra is sitting between Ahmed and Geetha and Balaji and Eshwar are not sitting opposite to each other. Who is third to

the left of Dhinesh?

29. If every alternative letter starting from B of the English alphabet is written in small letter, rest all are written in capital letters, how the month " September" be written.
(1) SeptEMbEr (2) SEpTeMBEr (3) SeptembeR
(4) SepteMber (5) None of the above.

30. The length of the side of a square is represented by x+2. The length of the side of an equilateral triangle is 2x. If the square and the equilateral triangle have equal perimeter, then the value of x is _____.

31. It takes Mr. Karthik y hours to complete typing a manuscript. After 2 hours, he was called away. What fractional part of the assignment was left incomplete?

32. Which of the following is larger than 3/5?
(1) ½ (2) 39/50 (3) 7/25 (4) 3/10 (5) 59/100

33. The number that does not have a reciprocal is _____.

34. There are 3 persons Sudhir, Arvind, and Gauri. Sudhir lent cars to Arvind and Gauri as many as they had already. After some time Arvind gave as many cars to Sudhir and Gauri as many as they have. After sometime Gauri did the same thing. At the end of this transaction each one of them had 24. Find the cars each originally had.

35. A man bought a horse and a cart. If he sold the horse at 10 % loss and the cart at 20 % gain, he would not lose anything; but if he sold the horse at 5% loss and the cart at 5% gain, he would lose Rs. 10 in the bargain. The amount paid by him was Rs._____ for the horse and Rs._____ for the cart.

Answers:

1. ***Answer:***
30 days.
Explanation:
Before:
One day work = 1 / 20
One man's one day work = 1 / (20 * 75)
Now:
No. Of workers = 50
One day work = 50 * 1 / (20 * 75)

The total no. of days required to complete the work = (75 * 20) / 50 = 30

2. ***Answer:***
0 %
Explanation:

Since 3x / 2 = x / (2 / 3)

3. ***Answer:***

5.3 %

Explanation:

He sells 950 grams of pulses and gains 50 grams.

If he sells 100 grams of pulses then he will gain (50 / 950) *100 = 5.26

4. ***Answer:***

250 lines of codes

5. ***Answer:***

7 days

Explanation:

The equation portraying the given problem is:

10 * x – 2 * (30 – x) = 216 where x is the number of working days.

Solving this we get x = 23

Number of days he was absent was 7 (30-23) days.

6. ***Answer:***

150 men.

Explanation:

One day's work = 2 / (7 * 90)

One hour's work = 2 / (7 * 90 * 8)

One man's work = 2 / (7 * 90 * 8 * 75)

The remaining work (5/7) has to be completed within 60 days, because the total number of days allotted for the project is 150 days.

So we get the equation

(2 * 10 * x * 60) / (7 * 90 * 8 * 75) = 5/7 where x is the number of men working after the 90^{th} day.

We get x = 225

Since we have 75 men already, it is enough to add only 150 men.

7. ***Answer:***

(c) 1

Explanation:

a percent of b : (a/100) * b

b percent of a : (b/100) * a

a percent of b divided by b percent of a : ((a / 100)*b) / (b/100) * a)) = 1

8. ***Answer:***

Cost price of horse = Rs. 400 & the cost price of cart = 200.

Explanation:-

Let x be the cost price of the horse and y be the cost price of the cart.

In the first sale there is no loss or profit. (i.e.) The loss obtained is equal to the gain.

Therefore (10/100) * x = (20/100) * y

X = 2 * y ----------(1)

In the second sale, he lost Rs. 10. (i.e.) The loss is greater than the profit by Rs. 10.

Therefore (5 / 100) * x = (5 / 100) * y + 10 -------(2)

Substituting (1) in (2) we get

(10 / 100) * y = (5 / 100) * y + 10

(5 / 100) * y = 10

y = 200

From (1) 2 * 200 = **x = 400**

9. ***Answer:***

3.

Explanation:

Since inclusion of any male player will reject a female from the team. Since there should be four member in the team and only three males are available, the girl, n should included in the team always irrespective of others selection.

10. ***Answer:***

5

11. ***Answer:***

1,2,3 & 4

12. ***Answer:***

B

13. ***Answer:***

11 & 9 apples per tree.

Explanation:

Let a, b, c, d & e be the total number of apples bored per year in A, B, C, D & E ‘s orchard. Given that a + 1 = b + 3 = c – 1 = d + 3 = e – 6

But the question is to find the number of apples bored per tree in C and D ‘s orchard. If is enough to consider c – 1 = d + 3.

Since the number of trees in C’s orchard is 11 and that of D’s orchard is 13. Let x and y be the number of apples bored per tree in C & d ‘s orchard respectively.

Therefore 11 x – 1 = 13 y + 3

By trial and error method, we get the value for x and y as 11 and 9

14. ***Answer:***

G.

Explanation:

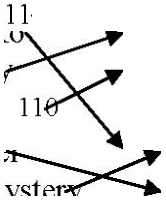
The order in which they are climbing is R – G – K – H – J

15 – 18

Answer:

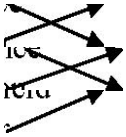
Novel Name Author Publisher

Spy thriller Rathko Heron



Murder mystery Gorky Piegon

Gothic romance Burchfield Blueja



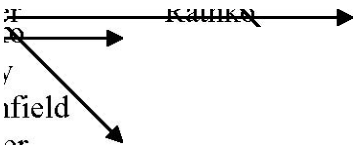
Science fiction Hopper Sparrow

Explanation:

Given

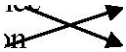
Novel Name Author Publisher

Spy thriller Rathko Heron



Murder mystery Gorky Piegon

Gothic romance Burchfield Blueja



Science fiction Hopper Sparrow

Since Blueja doesn't publish the novel by Burchfield and Heron publishes the novel spy thriller, Piegon publishes the novel by Burchfield.

Since Hopper writes Gothic romance and Heron publishes the novel spy thriller, Blueja publishes the novel by Hopper.

Since Heron publishes the novel spy thriller and Heron publishes the novel by Gorky, Gorky writes Spy thriller and Rathko writes Murder mystery.

19. ***Answer:***

451 times.

Explanation:

There are 60 minutes in an hour.

In ¾ of an hour there are (60 * ¾) minutes = 45 minutes.

In ¾ of an hour there are (60 * 45) seconds = 2700 seconds.

Light flashed for every 6 seconds.

In 2700 seconds 2700/6 = 450 times.

The count start after the first flash, the light will flashes 451 times in ¾ of an hour.

20. ***Answer:***

(4)

Explanation:

P

A B

Since p is a point on the line segment AB, AB > AP

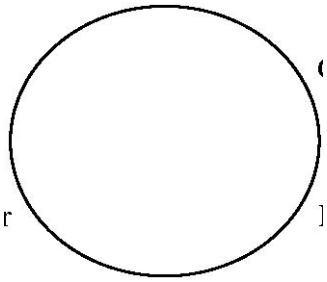
21. ***Answer:*** (c)

22. ***Answer:*** (b) & (d).

Ahmed

23 - 27.***Answer:***

Fakis Chandra



28. ***Answer:*** Fakis

Explanation: *Harsha* *Geetha*

Eswar Balaji

Dhinesh

29. ***Answer:***

(5).

Explanation:

Since every alternative letter starting from B of the English alphabet is written in small letter, the letters written in small letter are b, d, f...

In the first two answers the letter E is written in both small & capital letters, so they are not the correct answers. But in third and fourth answers the letter is written in small letter instead capital letter, so they are not the answers**.**

30. ***Answer:***

x = 4

Explanation:

Since the side of the square is x + 2, its perimeter = 4 (x + 2) = 4x + 8
Since the side of the equilateral triangle is 2x, its perimeter = 3 * 2x = 6x
Also, the perimeters of both are equal.
(i.e.) 4x + 8 = 6x
(i.e.) 2x = 8 x = 4.

31. ***Answer:***

(y – 2) / y.

Explanation:

To type a manuscript karthik took y hours.
Therefore his speed in typing = 1/y.
He was called away after 2 hours of typing.
Therefore the work completed = 1/y * 2.
Therefore the remaining work to be completed = 1 – 2/y.
(i.e.) work to be completed = (y-2)/y

32. ***Answer:***

(2)

33. ***Answer:***

1

Explanation:

One is the only number exists without reciprocal because the reciprocal of one is one itself.

34. ***Answer:***

Sudhir had 39 cars, Arvind had 21 cars and Gauri had 12 cars.

Explanation:

	Sudhir	Arvind	Gauri
Finally	24	24	24
Before Gauri's transaction	12	12	48
Before Arvind's transaction	6	42	24
Before Sudhir' s transaction	39	21	12

35. ***Answer:***

Cost price of horse: Rs. 400 &
Cost price of cart: Rs. 200

Explanation:

Let x be the cost of horse & y be the cost of the cart.
10 % of loss in selling horse = 20 % of gain in selling the cart
Therefore (10 / 100) * x = (20 * 100) * y
x = 2y ----------(1)

5 % of loss in selling the horse is 10 more than the 5 % gain in selling the cart.
Therefore (5 / 100) * x - 10 = (5 / 100) * y
5x - 1000 = 5y
Substituting (1)
10y - 1000 = 5y
5y = 1000
y = 200
x = 400 from (1)

Exercise 2.1

For the following, find the next term in the series

1. 6, 24, 60,120, 210

a) 336 b) 366 c) 330 d) 660

Answer : a) 336
Explanation : The series is 1.2.3, 2.3.4, 3.4.5, 4.5.6, 5.6.7, ... ('.' means product)

2. 1, 5, 13, 25

Answer : 41
Explanation : The series is of the form 0^2+1^2, 1^2+2^2,...

3. 0, 5, 8, 17

Answer : 24
Explanation : 1^2-1, 2^2+1, 3^2-1, 4^2+1, 5^2-1

4. 1, 8, 9, 64, 25 (Hint : Every successive terms are related)

Answer : 216
Explanation : 1^2, 2^3, 3^2, 4^3, 5^2, 6^3

5. 8,24,12,36,18,54

Answer : 27

6. 71,76,69,74,67,72
Answer : 67

7. 5,9,16,29,54
Answer : 103
Explanation : 5*2-1=9; 9*2-2=16; 16*2-3=29; 29*2-4=54; 54*2-5=103

8. 1,2,4,10,16,40,64 (Successive terms are related)
Answer : 200
Explanation : The series is powers of 2 (2^0,2^1,..).
All digits are less than 8. Every second number is in octal number system.
128 should follow 64. 128 base 10 = 200 base 8.

Exercise 2.2

Find the odd man out.

1. 3,5,7,12,13,17,19
Answer : 12
Explanation : All but 12 are odd numbers

2. 2,5,10,17,26,37,50,64
Answer : 64
Explanation : 2+3=5; 5+5=10; 10+7=17; 17+9=26; 26+11=37; 37+13=50; 50+15=65;

3. 105,85,60,30,0,-45,-90
Answer : 0
Explanation : 105-20=85; 85-25=60; 60-30=30; 30-35=-5; -5-40=-45; -45-45=-90;

Exercise 3
Solve the following.

1. What is the number of zeros at the end of the product of the numbers from 1 to 100?
Answer : 127
2. A fast typist can type some matter in 2 hours and a slow typist can type the same in 3 hours. If both type combinely, in how much time will they finish?
Answer : 1 hr 12 min
Explanation : The fast typist's work done in 1 hr = 1/2
The slow typist's work done in 1 hr = 1/3
If they work combinely, work done in 1 hr = 1/2+1/3 = 5/6
So, the work will be completed in 6/5 hours. i.e., 1+1/5 hours = 1hr 12 min

3. Gavaskar's average in his first 50 innings was 50. After the 51st innings, his average was 51.

How many runs did he score in his 51st innings. (supposing that he lost his wicket in his 51st innings)

Answer : 101

Explanation : Total score after 50 innings = 50*50 = 2500

Total score after 51 innings = 51*51 = 2601

So, runs made in the 51st innings = 2601-2500 = 101

If he had not lost his wicket in his 51st innings, he would have scored an unbeaten 50 in his 51st innings.

4. Out of 80 coins, one is counterfeit. What is the minimum number of weighings needed to find out the counterfeit coin?

Answer : 4

5. What can you conclude from the statement : All green are blue, all blue are red. ?

(i) some blue are green

(ii) some red are green

(iii) some green are not red

(iv) all red are blue

(a) i or ii but not both

(b) i & ii only

(c) iii or iv but not both

(d) iii & iv

Answer : (b)

6. A rectangular plate with length 8 inches, breadth 11 inches and thickness 2 inches is available. What is the length of the circular rod with diameter 8 inches and equal to the volume of the rectangular plate?

Answer : 3.5 inches

Explanation : Volume of the circular rod (cylinder) = Volume of the rectangular plate

(22/7)*4*4*h = 8*11*2

h = 7/2 = 3.5

7. What is the sum of all numbers between 100 and 1000 which are divisible by 14 ?

Answer : 35392

Explanation : The number closest to 100 which is greater than 100 and divisible by 14 is 112, which is the first term of the series which has to be summed.

The number closest to 1000 which is less than 1000 and divisible by 14 is 994, which is the last term of the series.

112 + 126 + ... + 994 = 14(8+9+ ... + 71) = 35392

8. If s(a) denotes square root of a, find the value of s(12+s(12+s(12+ ..... upto infinity.

Answer : 4

Explanation : Let x = s(12+s(12+s(12+.....

We can write x = s(12+x). i.e., x^2 = 12 + x. Solving this quadratic equation, we get

x = -3 or x=4. Sum cannot be -ve and hence sum = 4.

9. A cylindrical container has a radius of eight inches with a height of three inches. Compute how many inches should be added to either the radius or height to give the same increase in volume?

Answer : 16/3 inches

Explanation : Let x be the amount of increase. The volume will increase by the same amount if the radius increased or the height is increased.

So, the effect on increasing height is equal to the effect on increasing the radius.

i.e., (22/7)*8*8*(3+x) = (22/7)*(8+x)*(8+x)*3

Solving the quadratic equation we get the x = 0 or 16/3. The possible increase would be by 16/3 inches.

10. With just six weights and a balance scale, you can weigh any unit number of kgs from 1 to 364. What could be the six weights?

Answer : 1, 3, 9, 27, 81, 243 (All powers of 3)

11. Diophantus passed one sixth of his life in childhood, one twelfth in youth, and one seventh more as a bachelor; five years after his marriage a son was born who died four years before his father at half his final age. How old is Diophantus?

Answer : 84 years

Explanation : x/6 + x/12 + x/7 + 5 + x/2 + 4 = x

12 . If time at this moment is 9 P.M., what will be the time 23999999992 hours later?

Answer : 1 P.M.

Explanation : 24 billion hours later, it would be 9 P.M. and 8 hours before that it would be 1 P.M.

13. How big will an angle of one and a half degree look through a glass that magnifies things three times?

Answer : 1 1/2 degrees

Explanation : The magnifying glass cannot increase the magnitude of an angle.

14. Divide 45 into four parts such that when 2 is added to the first part, 2 is subtracted from the second part, 2 is multiplied by the third part and the fourth part is divided by two, all result in the same number.

Answer: 8, 12, 5, 20

Explanation: a + b + c + d =45; a+2 = b-2 = 2c = d/2; a=b-4; c = (b-2)/2; d = 2(b-2); b-4 + b + (b-2)/2 + 2(b-2) = 45;

15. I drove 60 km at 30 kmph and then an additional 60 km at 50 kmph. Compute my average speed over my 120 km.

Answer : 37 1/2

Explanation : Time reqd for the first 60 km = 120 min.; Time reqd for the second 60 km = 72 min.; Total time reqd = 192 min

Avg speed = (60*120)/192 = 37 1/2

Questions 16 and 17 are based on the following :

Five executives of European Corporation hold a Conference in Rome

Mr. A converses in Spanish & Italian
Mr. B, a spaniard, knows English also
Mr. C knows English and belongs to Italy
Mr. D converses in French and Spanish
Mr. E , a native of Italy knows French

16. Which of the following can act as interpreter if Mr. C & Mr. D wish to converse

a) only Mr. A b) Only Mr. B c) Mr. A & Mr. B d) Any of the other three

Answer : d) Any of the other three.

Explanation : From the data given, we can infer the following.

A knows Spanish, Italian
B knows Spanish, English
C knows Italian, English
D knows Spanish, French
E knows Italian, French

To act as an interpreter between C and D, a person has to know one of the combinations Italian&Spanish, Italian&French, English&Spanish, English&French

A, B, and E know atleast one of the combinations.

17. If a 6th executive is brought in, to be understood by maximum number of original five he should be fluent in

a) English & French b) Italian & Spanish c) English & French d) French & Italian

Answer : b) Italian & Spanish

Explanation : No of executives who know

i) English is 2
ii) Spanish is 3
iii) Italian is 3
iv) French is 2

Italian & Spanish are spoken by the maximum no of executives. So, if the 6th executive is fluent in Italian & Spanish, he can communicate with all the original five because everybody knows either Spanish or Italian.

18. What is the sum of the first 25 natural odd numbers?

Answer : 625

Explanation : The sum of the first n natural odd nos is square(n).

1+3 = 4 = square(2) 1+3+5 = 9 = square(3)

19. The sum of any seven consecutive numbers is divisible by

a) 2 b) 7 c) 3 d) 11

Exercise 3
Try the following.

1. There are seventy clerks working in a company, of which 30 are females. Also, 30 clerks are married; 24 clerks are above 25 years of age; 19 married clerks are above 25 years, of which 7 are males; 12 males are above 25 years of age; and 15 males are married. How many bachelor girls are there and how many of these are above 25?

2. A man sailed off from the North Pole. After covering 2,000 miles in one direction he turned West, sailed 2,000 miles, turned North and sailed ahead another 2,000 miles till he met his friend. How far was he from the North Pole and in what direction?

3. Here is a series of comments on the ages of three persons J, R, S by themselves.

   S : The difference between R's age and mine is three years.
   J : R is the youngest.
   R : Either I am 24 years old or J 25 or S 26.
   J : All are above 24 years of age.
   S : I am the eldest if and only if R is not the youngest.
   R : S is elder to me.
   J : I am the eldest.
   R : S is not 27 years old.
   S : The sum of my age and J's is two more than twice R's age.

   One of the three had been telling a lie throughout whereas others had spoken the truth. Determine the ages of S,J,R.

4. In a group of five people, what is the probability of finding two persons with the same month of birth?

5. A father and his son go out for a 'walk-and-run' every morning around a track formed by an equilateral triangle. The father's walking speed is 2 mph and his running speed is 5 mph. The son's walking and running speeds are twice that of his father. Both start together from one apex of the triangle, the son going clockwise and the father anti-clockwise. Initially the father runs and the son walks for a certain period of time. Thereafter, as soon as the father starts walking, the son starts running. Both complete the course in 45 minutes. For how long does the father run? Where do the two cross each other?

6. The Director of Medical Services was on his annual visit to the ENT Hospital. While going through the out patients' records he came across the following data for a particular day : " Ear consultations 45; Nose 50; Throat 70; Ear and Nose 30; Nose and Throat 20; Ear and Throat 30; Ear, Nose and Throat 10; Total patients 100." Then he came to the conclusion that the records were bogus. Was he right?

7. Amongst Ram, Sham and Gobind are a doctor, a lawyer and a police officer. They are

married to Radha, Gita and Sita (not in order). Each of the wives have a profession. Gobind's wife is an artist. Ram is not married to Gita. The lawyer's wife is a teacher. Radha is married to the police officer. Sita is an expert cook. Who's who?

8. What should come next?
   1, 2, 4, 10, 16, 40, 64,

Questions 9-12 are based on the following :

Three adults – Roberto, Sarah and Vicky – will be traveling in a van with five children – Freddy, Hillary, Jonathan, Lupe, and Marta. The van has a driver's seat and one passenger seat in the front, and two benches behind the front seats, one beach behind the other. Each bench has room for exactly three people. Everyone must sit in a seat or on a bench, and seating is subject to the following restrictions: An adult must sit on each bench.

Either Roberto or Sarah must sit in the driver's seat.
Jonathan must sit immediately beside Marta.

9. Of the following, who can sit in the front passenger seat ?
   (a) Jonathan (b) Lupe (c) Roberto (d) Sarah (e) Vicky

10. Which of the following groups of three can sit together on a bench?
    (a) Freddy, Jonathan and Marta (b) Freddy, Jonathan and Vicky
    (c) Freddy, Sarah and Vicky (d) Hillary, Lupe and Sarah
    (e) Lupe, Marta and Roberto

11. If Freddy sits immediately beside Vicky, which of the following cannot be true ?
    a. Jonathan sits immediately beside Sarah
    b. Lupe sits immediately beside Vicky
    c. Hillary sits in the front passenger seat
    d. Freddy sits on the same bench as Hillary
    e. Hillary sits on the same bench as Roberto

12. If Sarah sits on a bench that is behind where Jonathan is sitting, which of the following must be true ?
    a. Hillary sits in a seat or on a bench that is in front of where Marta is sitting
    b. Lupe sits in a seat or on a bench that is in front of where Freddy is sitting
    c. Freddy sits on the same bench as Hillary
    d. Lupe sits on the same bench as Sarah
    e. Marta sits on the same bench as Vicky

13. Make six squares of the same size using twelve match-sticks. (Hint : You will need an adhesive to arrange the required figure)

14. A farmer has two rectangular fields. The larger field has twice the length and 4 times the

width of the smaller field. If the smaller field has area K, then the are of the larger field is greater than the area of the smaller field by what amount?
(a) 6K (b) 8K (c) 12K (d) 7K

15. Nine equal circles are enclosed in a square whose area is 36sq units. Find the area of each circle.

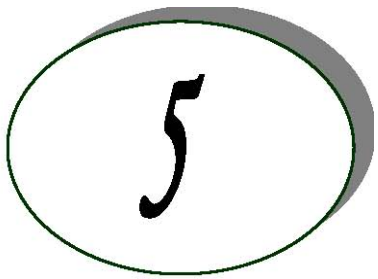
16. There are 9 cards. Arrange them in a 3*3 matrix. Cards are of 4 colors. They are red, yellow, blue, green. Conditions for arrangement: one red card must be in first row or second row. 2 green cards should be in 3^{rd} column. Yellow cards must be in the 3 corners only. Two blue cards must be in the 2nd row. At least one green card in each row.

17. Is z less than w? z and w are real numbers.
    (I) $z^2 = 25$
    (II) $w = 9$
    To answer the question,
    a) Either I or II is sufficient
    b) Both I and II are sufficient but neither of them is alone sufficient
    c) I & II are sufficient
    d) Both are not sufficient

18. A speaks truth 70% of the time; B speaks truth 80% of the time. What is the probability that both are contradicting each other?

19. In a family 7 children don't eat spinach, 6 don't eat carrot, 5 don't eat beans, 4 don't eat spinach & carrots, 3 don't eat carrot & beans, 2 don't eat beans & spinach. One doesn't eat all 3. Find the no. of children.

20. Anna, Bena, Catherina and Diana are at their monthly business meeting. Their occupations are author, biologist, chemist and doctor, but not necessarily in that order. Diana just told the neighbour, who is a biologist that Catherina was on her way with doughnuts. Anna is sitting across from the doctor and next to the chemist. The doctor was thinking that Bena was a good name for parent's to choose, but didn't say anything. What is each person's occupation?



SECTION - I

FILE MANAGEMENT IN UNIX

1. How are devices represented in UNIX?

All devices are represented by files called *special files* that are located in/dev directory. Thus, device files and other files are named and accessed in the same way. A 'regular file' is just an ordinary data file in the disk. A 'block special file' represents a device with characteristics similar to a disk (data transfer in terms of blocks). A 'character special file' represents a device with characteristics similar to a keyboard (data transfer is by stream of bits in sequential order).

2. What is 'inode'?

All UNIX files have its description stored in a structure called 'inode'. The inode contains info about the file-size, its location, time of last access, time of last modification, permission and so on. Directories are also represented as files and have an associated inode. In addition to descriptions about the file, the inode contains pointers to the data blocks of the file. If the file is large, inode has indirect pointer to a block of pointers to additional data blocks (this further aggregates for larger files). A block is typically 8k.

Inode consists of the following fields:

File owner identifier
File type
File access permissions
File access times
Number of links
File size
Location of the file data

3. Brief about the directory representation in UNIX

A Unix directory is a file containing a correspondence between filenames and inodes. A directory is a special file that the kernel maintains. Only kernel modifies directories, but processes can read directories. The contents of a directory are a list of filename and inode number pairs. When new directories are created, kernel makes two entries named '.' (refers to the directory itself) and '..' (refers to parent directory).

System call for creating directory is mkdir (pathname, mode).

4. What are the Unix system calls for I/O?

open(pathname,flag,mode) - open file
creat(pathname,mode) - create file

close(filedes) - close an open file
read(filedes,buffer,bytes) - read data from an open file
write(filedes,buffer,bytes) - write data to an open file
lseek(filedes,offset,from) - position an open file
dup(filedes) - duplicate an existing file descriptor
dup2(oldfd,newfd) - duplicate to a desired file descriptor
fcntl(filedes,cmd,arg) - change properties of an open file
ioctl(filedes,request,arg) - change the behaviour of an open file

The difference between fcntl anf ioctl is that the former is intended for any open file, while the latter is for device-specific operations.

5. How do you change File Access Permissions?

Every file has following attributes:
owner's user ID (16 bit integer)
owner's group ID (16 bit integer)
File access mode word

'r w x -r w x- r w x'
(user permission-group permission-others permission)
r-read, w-write, x-execute
To change the access mode, we use chmod(filename,mode).
Example 1:

To change mode of myfile to 'rw-rw-r--' (ie. read, write permission for user - read,write permission for group - only read permission for others) we give the args as:

chmod(myfile,0664) .

Each operation is represented by discrete values

'r' is 4
'w' is 2
'x' is 1

Therefore, for 'rw' the value is 6(4+2).
Example 2:

To change mode of myfile to 'rwxr--r--' we give the args as:
chmod(myfile,0744).

6. What are links and symbolic links in UNIX file system?

A link is a second name (not a file) for a file. Links can be used to assign more than one name to a file, but cannot be used to assign a directory more than one name or link filenames on different computers.

Symbolic link 'is' a file that only contains the name of another file.Operation on the symbolic link is directed to the file pointed by the it.Both the limitations of links are eliminated in symbolic links.

Commands for linking files are:
Link ln filename1 filename2
Symbolic link ln -s filename1 filename2

7. What is a FIFO?

FIFO are otherwise called as 'named pipes'. FIFO (first-in-first-out) is a special file which is said to be data transient. Once data is read from named pipe, it cannot be read again. Also, data can be read only in the order written. It is used in interprocess communication where a process writes to one end of the pipe (producer) and the other reads from the other end (consumer).

8. How do you create special files like named pipes and device files?

The system call mknod creates special files in the following sequence.

1. kernel assigns new inode,
2. sets the file type to indicate that the file is a pipe, directory or special file,
3. If it is a device file, it makes the other entries like major, minor device numbers.

For example:

If the device is a disk, major device number refers to the disk controller and minor device number is the disk.

9. Discuss the mount and unmount system calls

The privileged mount system call is used to attach a file system to a directory of another file system; the unmount system call detaches a file system. When you mount another file system on to your directory, you are essentially splicing one directory tree onto a branch in another directory tree. The first argument to mount call is the mount point, that is , a directory in the current file naming system. The second argument is the file system to mount to that point. When you insert a cdrom to your unix system's drive, the file system in the cdrom automatically mounts to /dev/cdrom in your system.

10. How does the inode map to data block of a file?

Inode has 13 block addresses. The first 10 are direct block addresses of the first 10 data blocks in the file. The 11th address points to a one-level index block. The 12th address points to a two-level (double in-direction) index block. The 13th address points to a three-level(triple in-direction)index block. This provides a very large maximum file size with efficient access to large files, but also small files are accessed directly in one disk read.

11. What is a shell?

A shell is an interactive user interface to an operating system services that allows an user to enter commands as character strings or through a graphical user interface. The shell converts them to system calls to the OS or forks off a process to execute the command. System call results and other information from the OS are presented to the user through an interactive interface. Commonly used shells are sh,csh,ks etc.

SECTION - II

PROCESS MODEL and IPC

1. Brief about the initial process sequence while the system boots up.

While booting, special process called the 'swapper' or 'scheduler' is created with Process-ID 0. The swapper manages memory allocation for processes and influences CPU

allocation. The swapper inturn creates 3 children:
the process dispatcher,
vhand and
dbflush

with IDs 1,2 and 3 respectively.

This is done by executing the file /etc/init. Process dispatcher gives birth to the shell. Unix keeps track of all the processes in an internal data structure called the Process Table (listing command is ps -el).

2. What are various IDs associated with a process?

Unix identifies each process with a unique integer called ProcessID. The process that executes the request for creation of a process is called the 'parent process' whose PID is 'Parent Process ID'. Every process is associated with a particular user called the 'owner' who has privileges over the process. The identification for the user is 'UserID'. Owner is the user who executes the process. Process also has 'Effective User ID' which determines the access privileges for accessing resources like files.

getpid() -process id
getppid() -parent process id
getuid() -user id
geteuid() -effective user id

3. Explain fork() system call.

The `fork()' used to create a new process from an existing process. The new process is called the child process, and the existing process is called the parent. We can tell which is which by checking the return value from `fork()'. The parent gets the child's pid returned to him, but the child gets 0 returned to him.

4. Predict the output of the following program code

```
main()
{
  fork();
        printf("Hello World!");
}
```

Answer:

Hello World!Hello World!

Explanation:

The fork creates a child that is a duplicate of the parent process. The child begins from the fork().All the statements after the call to fork() will be executed twice.(once by the parent process and other by child). The statement before fork() is executed only by the parent process.

5. Predict the output of the following program code

```
main()
{
        fork(); fork(); fork();
        printf("Hello World!");
```

}

Answer:

"Hello World" will be printed 8 times.

Explanation:

2^n times where n is the number of calls to fork()

6. List the system calls used for process management:

System calls	Description
fork()	To create a new process
exec()	To execute a new program in a process
wait()	To wait until a created process completes its execution
exit()	To exit from a process execution
getpid()	To get a process identifier of the current process
getppid()	To get parent process identifier
nice()	To bias the existing priority of a process
brk()	To increase/decrease the data segment size of a process

7. How can you get/set an environment variable from a program?

Getting the value of an environment variable is done by using `getenv()'.

Setting the value of an environment variable is done by using `putenv()'.

8. How can a parent and child process communicate?

A parent and child can communicate through any of the normal inter-process communication schemes (pipes, sockets, message queues, shared memory), but also have some special ways to communicate that take advantage of their relationship as a parent and child. One of the most obvious is that the parent can get the exit status of the child.

9. What is a zombie?

When a program forks and the child finishes before the parent, the kernel still keeps some of its information about the child in case the parent might need it - for example, the parent may need to check the child's exit status. To be able to get this information, the parent calls `wait()'; In the interval between the child terminating and the parent calling `wait()', the child is said to be a `zombie' (If you do `ps', the child will have a `Z' in its status field to indicate this.)

10. What are the process states in Unix?

As a process executes it changes state according to its circumstances. Unix processes have the following states:

Running : The process is either running or it is ready to run .

Waiting : The process is waiting for an event or for a resource.

Stopped : The process has been stopped, usually by receiving a signal.

Zombie : The process is dead but have not been removed from the process table.

11. What Happens when you execute a program?

When you execute a program on your UNIX system, the system creates a special environment for that program. This environment contains everything needed for the system to run the program as if no other program were running on the system. Each process has process

context, which is everything that is unique about the state of the program you are currently running. Every time you execute a program the UNIX system does a fork, which performs a series of operations to create a process context and then execute your program in that context. The steps include the following:

Allocate a slot in the process table, a list of currently running programs kept by UNIX.
Assign a unique process identifier (PID) to the process.
iCopy the context of the parent, the process that requested the spawning of the new process.
Return the new PID to the parent process. This enables the parent process to examine or control the process directly.

After the fork is complete, UNIX runs your program.

12. What Happens when you execute a command?

When you enter 'ls' command to look at the contents of your current working directory, UNIX does a series of things to create an environment for ls and the run it: The shell has UNIX perform a fork. This creates a new process that the shell will use to run the ls program. The shell has UNIX perform an exec of the ls program. This replaces the shell program and data with the program and data for ls and then starts running that new program. The ls program is loaded into the new process context, replacing the text and data of the shell. The ls program performs its task, listing the contents of the current directory.

13. What is a Daemon?

A daemon is a process that detaches itself from the terminal and runs, disconnected, in the background, waiting for requests and responding to them. It can also be defined as the background process that does not belong to a terminal session. Many system functions are commonly performed by daemons, including the sendmail daemon, which handles mail, and the NNTP daemon, which handles USENET news. Many other daemons may exist. Some of the most common daemons are:

init: Takes over the basic running of the system when the kernel has finished the boot process.
inetd: Responsible for starting network services that do not have their own stand-alone daemons. For example, inetd usually takes care of incoming rlogin, telnet, and ftp connections.
cron: Responsible for running repetitive tasks on a regular schedule.

14. What is 'ps' command for?

The ps command prints the process status for some or all of the running processes. The information given are the process identification number (PID),the amount of time that the process has taken to execute so far etc.

15. How would you kill a process?

The kill command takes the PID as one argument; this identifies which process to terminate. The PID of a process can be got using 'ps' command.

16. What is an advantage of executing a process in background?

The most common reason to put a process in the background is to allow you to do

something else interactively without waiting for the process to complete. At the end of the command you add the special background symbol, &. This symbol tells your shell to execute the given command in the background.

Example: cp *.* ../backup& (cp is for copy)

17. How do you execute one program from within another?

The system calls used for low-level process creation are execlp() and execvp(). The execlp call overlays the existing program with the new one , runs that and exits. The original program gets back control only when an error occurs.

execlp(path,file_name,arguments..); //last argument must be NULL

A variant of execlp called execvp is used when the number of arguments is not known in advance.

execvp(path,argument_array); //argument array should be terminated by NULL

18. What is IPC? What are the various schemes available?

The term IPC (Inter-Process Communication) describes various ways by which different process running on some operating system communicate between each other. Various schemes available are as follows:

Pipes:

One-way communication scheme through which different process can communicate. The problem is that the two processes should have a common ancestor (parent-child relationship). However this problem was fixed with the introduction of named-pipes (FIFO).

Message Queues :

Message queues can be used between related and unrelated processes running on a machine.

Shared Memory:

This is the fastest of all IPC schemes. The memory to be shared is mapped into the address space of the processes (that are sharing). The speed achieved is attributed to the fact that there is no kernel involvement. But this scheme needs synchronization.

Various forms of synchronisation are mutexes, condition-variables, read-write locks, record-locks, and semaphores.

SECTION - III

MEMORY MANAGEMENT

1. What is the difference between Swapping and Paging?

Swapping:

Whole process is moved from the swap device to the main memory for execution. Process size must be less than or equal to the available main memory. It is easier to implementation and overhead to the system. Swapping systems does not handle the memory more flexibly as compared to the paging systems.

Paging:

Only the required memory pages are moved to main memory from the swap device for execution. Process size does not matter. Gives the concept of the virtual memory.

It provides greater flexibility in mapping the virtual address space into the physical memory of the machine. Allows more number of processes to fit in the main memory simultaneously. Allows the greater process size than the available physical memory. Demand paging systems handle the memory more flexibly.

2. What is major difference between the Historic Unix and the new BSD release of Unix System V in terms of Memory Management?

Historic Unix uses Swapping – entire process is transferred to the main memory from the swap device, whereas the Unix System V uses Demand Paging – only the part of the process is moved to the main memory. Historic Unix uses one Swap Device and Unix System V allow multiple Swap Devices.

3. What is the main goal of the Memory Management?

It decides which process should reside in the main memory,
Manages the parts of the virtual address space of a process which is non-core resident,
Monitors the available main memory and periodically write the processes into the swap device to provide more processes fit in the main memory simultaneously.

4. What is a Map?

A Map is an Array, which contains the addresses of the free space in the swap device that are allocatable resources, and the number of the resource units available there.

This allows First-Fit allocation of contiguous blocks of a resource. Initially the Map contains one entry – address (block offset from the starting of the swap area) and the total number of resources.

Kernel treats each unit of Map as a group of disk blocks. On the allocation and freeing of the resources Kernel updates the Map for accurate information.

5. What scheme does the Kernel in Unix System V follow while choosing a swap device among the multiple swap devices?

Kernel follows Round Robin scheme choosing a swap device among the multiple swap devices in Unix System V.

6. What is a Region?

A Region is a continuous area of a process's address space (such as text, data and stack). The kernel in a 'Region Table' that is local to the process maintains region. Regions are sharable among the process.

7. What are the events done by the Kernel after a process is being swapped out from the main memory?

When Kernel swaps the process out of the primary memory, it performs the following:

Kernel decrements the Reference Count of each region of the process. If the reference count becomes zero, swaps the region out of the main memory,

Kernel allocates the space for the swapping process in the swap device,

Kernel locks the other swapping process while the current swapping operation is going on,

The Kernel saves the swap address of the region in the region table.

8. Is the Process before and after the swap are the same? Give reason.

Process before swapping is residing in the primary memory in its original form. The regions (text, data and stack) may not be occupied fully by the process, there may be few empty slots in any of the regions and while swapping Kernel do not bother about the empty slots while swapping the process out.

After swapping the process resides in the swap (secondary memory) device. The regions swapped out will be present but only the occupied region slots but not the empty slots that were present before assigning.

While swapping the process once again into the main memory, the Kernel referring to the Process Memory Map, it assigns the main memory accordingly taking care of the empty slots in the regions.

9. What do you mean by u-area (user area) or u-block?

This contains the private data that is manipulated only by the Kernel. This is local to the Process, i.e. each process is allocated a u-area.

10. What are the entities that are swapped out of the main memory while swapping the process out of the main memory?

All memory space occupied by the process, process's u-area, and Kernel stack are swapped out, theoretically.

Practically, if the process's u-area contains the Address Translation Tables for the process then Kernel implementations do not swap the u-area.

11. What is Fork swap?

fork() is a system call to create a child process. When the parent process calls fork() system call, the child process is created and if there is short of memory then the child process is sent to the read-to-run state in the swap device, and return to the user state without swapping the parent process. When the memory will be available the child process will be swapped into the main memory.

12. What is Expansion swap?

Address	Units
1	10,000

At the time when any process requires more memory than it is currently allocated, the Kernel performs Expansion swap. To do this Kernel reserves enough space in the swap device. Then the address translation mapping is adjusted for the new virtual address space but the physical

memory is not allocated. At last Kernel swaps the process into the assigned space in the swap device. Later when the Kernel swaps the process into the main memory this assigns memory according to the new address translation mapping.

13. How the Swapper works?

The swapper is the only process that swaps the processes. The Swapper operates only in the Kernel mode and it does not uses System calls instead it uses internal Kernel functions for swapping. It is the archetype of all kernel process.

Kernel contains 4 data structures for Demand paging. They are,

Page table entries,
Disk block descriptors,
Page frame data table (pfdata),
Swap-use table.

26. What are the bits that support the demand paging?

Valid, Reference, Modify, Copy on write, Age. These bits are the part of the page table entry, which includes physical address of the page and protection bits.

Page address	Age	Copy on write	Modify	Reference	Valid	Protection

27. How the Kernel handles the fork() system call in traditional Unix and in the System V Unix, while swapping?

Kernel in traditional Unix, makes the duplicate copy of the parent's address space and attaches it to the child's process, while swapping. Kernel in System V Unix, manipulates the region tables, page table, and pfdata table entries, by incrementing the reference count of the region table of shared regions.

28. Difference between the fork() and vfork() system call?

During the fork() system call the Kernel makes a copy of the parent process's address space and attaches it to the child process.

But the vfork() system call do not makes any copy of the parent's address space, so it is faster than the fork() system call. The child process as a result of the vfork() system call executes exec() system call. The child process from vfork() system call executes in the parent's address space (this can overwrite the parent's data and stack) which suspends the parent process until the child process exits.

29. What is BSS(Block Started by Symbol)?

A data representation at the machine level, that has initial values when a program starts and tells about how much space the kernel allocates for the un-initialized data. Kernel initializes it to zero at run-time.

30. What is Page-Stealer process?

This is the Kernel process that makes rooms for the incoming pages, by swapping the memory pages that are not the part of the working set of a process. Page-Stealer is created by the Kernel at the system initialization and invokes it throughout the lifetime of the system. Kernel locks a region when a process faults on a page in the region, so that page stealer cannot steal the page, which is being faulted in.

31. Name two paging states for a page in memory?

The two paging states are:

The page is aging and is not yet eligible for swapping,

The page is eligible for swapping but not yet eligible for reassignment to other virtual address space.

32. What are the phases of swapping a page from the memory?

Page stealer finds the page eligible for swapping and places the page number in the list of pages to be swapped.

Kernel copies the page to a swap device when necessary and clears the *valid* bit in the page table entry, decrements the pfdata reference count, and places the pfdata table entry at the end of the free list if its reference count is 0.

33. What is page fault? Its types?

Page fault refers to the situation of not having a page in the main memory when any process references it.

There are two types of page fault :

Validity fault,

Protection fault.

34. In what way the Fault Handlers and the Interrupt handlers are different?

Fault handlers are also an interrupt handler with an exception that the interrupt handlers cannot sleep. Fault handlers sleep in the context of the process that caused the memory fault. The fault refers to the running process and no arbitrary processes are put to sleep.

35. What is validity fault?

If a process referring a page in the main memory whose valid bit is not set, it results in validity fault.

The valid bit is not set for those pages:

that are outside the virtual address space of a process,

that are the part of the virtual address space of the process but no physical address is assigned to it.

36. What does the swapping system do if it identifies the illegal page for swapping?

If the disk block descriptor does not contain any record of the faulted page, then this causes the attempted memory reference is invalid and the kernel sends a *"Segmentation violation"* signal to the offending process. This happens when the swapping system identifies any invalid memory reference.

37. What are states that the page can be in, after causing a page fault?

On a swap device and not in memory,
On the free page list in the main memory,
In an executable file,
Marked "demand zero",
Marked "demand fill".

38. *In what way the validity fault handler concludes?*

It sets the valid bit of the page by clearing the modify bit.
It recalculates the process priority.

39. *At what mode the fault handler executes?*

At the Kernel Mode.

40. *What do you mean by the protection fault?*

Protection fault refers to the process accessing the pages, which do not have the access permission. A process also incur the protection fault when it attempts to write a page whose *copy on write* bit was set during the fork() system call.

41. *How the Kernel handles the copy on write bit of a page, when the bit is set?*

In situations like, where the copy on write bit of a page is set and that page is shared by more than one process, the Kernel allocates new page and copies the content to the new page and the other processes retain their references to the old page. After copying the Kernel updates the page table entry with the new page number. Then Kernel decrements the reference count of the old pfdata table entry.

In cases like, where the copy on write bit is set and no processes are sharing the page, the Kernel allows the physical page to be reused by the processes. By doing so, it clears the copy on write bit and disassociates the page from its disk copy (if one exists), because other process may share the disk copy. Then it removes the pfdata table entry from the page-queue as the new copy of the virtual page is not on the swap device. It decrements the swap-use count for the page and if count drops to 0, frees the swap space.

42. *For which kind of fault the page is checked first?*

The page is first checked for the validity fault, as soon as it is found that the page is invalid (valid bit is clear), the validity fault handler returns immediately, and the process incur the

validity page fault. Kernel handles the validity fault and the process will incur the protection fault if any one is present.

43. *In what way the protection fault handler concludes?*

After finishing the execution of the fault handler, it sets the *modify* and *protection* bits and clears the *copy on write* bit. It recalculates the process-priority and checks for signals.

44. *How the Kernel handles both the page stealer and the fault handler?*

The page stealer and the fault handler thrash because of the shortage of the memory. If the sum of the working sets of all processes is greater that the physical memory then the fault handler will usually sleep because it cannot allocate pages for a process. This results in the reduction of the system throughput because Kernel spends too much time in overhead, rearranging the memory in the frantic pace.

1. What is database?

A database is a logically coherent collection of data with some inherent meaning, representing some aspect of real world and which is designed, built and populated with data for a specific purpose.

2. What is DBMS?

It is a collection of programs that enables user to create and maintain a database. In other words it is general-purpose software that provides the users with the processes of *defining*, *constructing* and *manipulating* the database for various applications.

3. What is a Database system?

The database and DBMS software together is called as Database system.

4. Advantages of DBMS?

Redundancy is controlled.
Unauthorised access is restricted.
Providing multiple user interfaces.
Enforcing integrity constraints.
Providing backup and recovery.

5. Disadvantage in File Processing System?

Data redundancy & inconsistency.

Difficult in accessing data.
Data isolation.
Data integrity.
Concurrent access is not possible.
Security Problems.

6. Describe the three levels of data abstraction?

The are three levels of abstraction:

Physical level: The lowest level of abstraction describes how data are stored.

Logical level: The next higher level of abstraction, describes what data are stored in database and what relationship among those data.

View level: The highest level of abstraction describes only part of entire database.

7. Define the "integrity rules"

There are two Integrity rules.

Entity Integrity: States that “Primary key cannot have NULL value”

Referential Integrity: States that “Foreign Key can be either a NULL value or should be Primary Key value of other relation.

8. What is extension and intension?

Extension -

It is the number of tuples present in a table at any instance. This is time dependent.

Intension -

It is a constant value that gives the name, structure of table and the constraints laid on it.

9. What is System R? What are its two major subsystems?

System R was designed and developed over a period of 1974-79 at IBM San Jose Research Center. It is a prototype and its purpose was to demonstrate that it is possible to build a Relational System that can be used in a real life environment to solve real life problems, with performance at least comparable to that of existing system.

Its two subsystems are

Research Storage

System Relational Data System.

10. How is the data structure of System R different from the relational structure?

Unlike Relational systems in System R

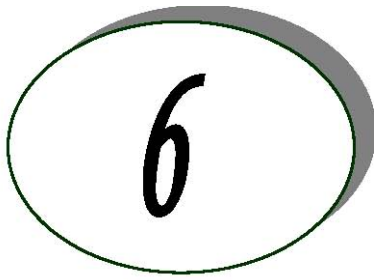
Domains are not supported

Enforcement of candidate key uniqueness is optional

Enforcement of entity integrity is optional

Referential integrity is not enforced

11. What is Data Independence?



RDBMS Concepts

Data independence means that "the application is independent of the storage structure and access strategy of data". In other words, The ability to modify the schema definition in one level should not affect the schema definition in the next higher level.

Two types of Data Independence:

Physical Data Independence: Modification in physical level should not affect the logical level.

Logical Data Independence: Modification in logical level should affect the view level.

NOTE: Logical Data Independence is more difficult to achieve

12. What is a view? How it is related to data independence?

A view may be thought of as a virtual table, that is, a table that does not really exist in its own right but is instead derived from one or more underlying base table. In other words, there is no stored file that direct represents the view instead a definition of view is stored in data dictionary.

Growth and restructuring of base tables is not reflected in views. Thus the view can insulate users from the effects of restructuring and growth in the database. Hence accounts for logical data independence.

13. What is Data Model?

A collection of conceptual tools for describing data, data relationships data semantics and constraints.

14. What is E-R model?

This data model is based on real world that consists of basic objects called entities and of relationship among these objects. Entities are described in a database by a set of attributes.

15. What is Object Oriented model?

This model is based on collection of objects. An object contains values stored in instance variables with in the object. An object also contains bodies of code that operate on the object. These bodies of code are called methods. Objects that contain same types of values and the same methods are grouped together into classes.

16. What is an Entity?

It is a 'thing' in the real world with an independent existence.

17. What is an Entity type?

It is a collection (set) of entities that have same attributes.

18. What is an Entity set?

It is a collection of all entities of particular entity type in the database.

19. What is an Extension of entity type?

The collections of entities of a particular entity type are grouped together into an entity set.

20. What is Weak Entity set?

An entity set may not have sufficient attributes to form a primary key, and its primary key compromises of its partial key and primary key of its parent entity, then it is said to be Weak Entity set.

21. What is an attribute?

It is a particular property, which describes the entity.

22. What is a Relation Schema and a Relation?

A relation Schema denoted by R(A1, A2, …, An) is made up of the relation name R and the list of attributes Ai that it contains. A relation is defined as a set of tuples. Let r be the relation which contains set tuples (t1, t2, t3, ..., tn). Each tuple is an ordered list of n-values t=(v1,v2, ..., vn).

23. What is degree of a Relation?

It is the number of attribute of its relation schema.

24. What is Relationship?

It is an association among two or more entities.

25. What is Relationship set?

The collection (or set) of similar relationships.

26. What is Relationship type?

Relationship type defines a set of associations or a relationship set among a given set of entity types.

27. What is degree of Relationship type?

It is the number of entity type participating.

25. What is DDL (Data Definition Language)?

A data base schema is specifies by a set of definitions expressed by a special language called DDL.

26. What is VDL (View Definition Language)?

It specifies user views and their mappings to the conceptual schema.

27. What is SDL (Storage Definition Language)?

This language is to specify the internal schema. This language may specify the mapping between two schemas.

28. What is Data Storage - Definition Language?

The storage structures and access methods used by database system are specified by a set of definition in a special type of DDL called data storage-definition language.

29. What is DML (Data Manipulation Language)?

This language that enable user to access or manipulate data as organised by appropriate data model.

Procedural DML or Low level: DML requires a user to specify what data are needed and how to get those data.

Non-Procedural DML or High level: DML requires a user to specify what data are needed without specifying how to get those data.

31. What is DML Compiler?

It translates DML statements in a query language into low-level instruction that the query evaluation engine can understand.

32. What is Query evaluation engine?

It executes low-level instruction generated by compiler.

33. What is DDL Interpreter?

It interprets DDL statements and record them in tables containing metadata.

34. What is Record-at-a-time?

The Low level or Procedural DML can specify and retrieve each record from a set of records. This retrieve of a record is said to be Record-at-a-time.

35. What is Set-at-a-time or Set-oriented?

The High level or Non-procedural DML can specify and retrieve many records in a single DML statement. This retrieve of a record is said to be Set-at-a-time or Set-oriented.

36. What is Relational Algebra?

It is procedural query language. It consists of a set of operations that take one or two relations as input and produce a new relation.

37. What is Relational Calculus?

It is an applied predicate calculus specifically tailored for relational databases proposed by E.F. Codd. E.g. of languages based on it are DSL ALPHA, QUEL.

38. How does Tuple-oriented relational calculus differ from domain-oriented relational calculus

The tuple-oriented calculus uses a tuple variables i.e., variable whose only permitted values are tuples of that relation. E.g. QUEL

The domain-oriented calculus has domain variables i.e., variables that range over the underlying domains instead of over relation. E.g. ILL, DEDUCE.

39. What is normalization?

It is a process of analysing the given relation schemas based on their Functional Dependencies (FDs) and primary key to achieve the properties

Minimizing redundancy

Minimizing insertion, deletion and update anomalies.

40. What is Functional Dependency?

A Functional dependency is denoted by X Y between two sets of attributes X and Y that are subsets of R specifies a constraint on the possible tuple that can form a relation state r of R. The constraint is for any two tuples t1 and t2 in r if t1[X] = t2[X] then they have t1[Y] = t2[Y]. This means the value of X component of a tuple uniquely determines the value of component Y.

41. When is a functional dependency F said to be minimal?

Every dependency in F has a single attribute for its right hand side.

We cannot replace any dependency X A in F with a dependency Y A where Y is a proper subset of X and still have a set of dependency that is equivalent to F.

We cannot remove any dependency from F and still have set of dependency that is equivalent to F.

42. What is Multivalued dependency?

Multivalued dependency denoted by X Y specified on relation schema R, where X and Y are both subsets of R, specifies the following constraint on any relation r of R: if two tuples t1 and t2 exist in r such that t1[X] = t2[X] then t3 and t4 should also exist in r with the following properties

t3[x] = t4[X] = t1[X] = t2[X]

t3[Y] = t1[Y] and t4[Y] = t2[Y]

t3[Z] = t2[Z] and t4[Z] = t1[Z]

where [Z = (R-(X U Y))]

43. What is Lossless join property?

It guarantees that the spurious tuple generation does not occur with respect to relation schemas after decomposition.

44. What is 1 NF (Normal Form)?

The domain of attribute must include only atomic (simple, indivisible) values.

45. What is Fully Functional dependency?

It is based on concept of full functional dependency. A functional dependency X Y is full functional dependency if removal of any attribute A from X means that the dependency does not hold any more.

46. What is 2NF?

A relation schema R is in 2NF if it is in 1NF and every non-prime attribute A in R is fully functionally dependent on primary key.

47. What is 3NF?

A relation schema R is in 3NF if it is in 2NF and for every FD X A either of the following is true

X is a Super-key of R.

A is a prime attribute of R.

In other words, if every non prime attribute is non-transitively dependent on primary key.

48. What is BCNF (Boyce-Codd Normal Form)?

→

↔

A relation schema R is in BCNF if it is in 3NF and satisfies an additional constraint that for every FD X A, X must be a candidate key.

→

49. What is 4NF?

A relation schema R is said to be in 4NF if for every Multivalued dependency X Y that holds over R, one of following is true

X is subset or equal to (or) XY = R.

X is a super key.

50. What is 5NF?

A Relation schema R is said to be 5NF if for every join dependency {R1, R2, ..., Rn} that holds R, one the following is true

Ri = R for some i.

The join dependency is implied by the set of FD, over R in which the left side is key of R.

51. What is Domain-Key Normal Form?

A relation is said to be in DKNF if all constraints and dependencies that should hold on the the constraint can be enforced by simply enforcing the domain constraint and key constraint on the relation.

→

↠

52. What are partial, alternate,, artificial, compound and natural key?

Partial Key:

It is a set of attributes that can uniquely identify weak entities and that are related to same owner entity. It is sometime called as Discriminator.

Alternate Key:

All Candidate Keys excluding the Primary Key are known as Alternate Keys.

Artificial Key:

If no obvious key, either stand alone or compound is available, then the last resort is to simply create a key, by assigning a unique number to each record or occurrence. Then this is known as developing an artificial key.

*Compound Key**:***

If no single data element uniquely identifies occurrences within a construct, then combining multiple elements to create a unique identifier for the construct is known as creating a compound key.

*Natural Key**:***

When one of the data elements stored within a construct is utilized as the primary key, then it is called the natural key.

53. What is indexing and what are the different kinds of indexing?

Indexing is a technique for determining how quickly specific data can be found.

Types:

Binary search style indexing
B-Tree indexing
Inverted list indexing
Memory resident table
Table indexing

54. What is system catalog or catalog relation? How is better known as?

A RDBMS maintains a description of all the data that it contains, information about every relation and index that it contains. This information is stored in a collection of relations maintained by the system called metadata. It is also called data dictionary.

55. What is meant by query optimization?

The phase that identifies an efficient execution plan for evaluating a query that has the least estimated cost is referred to as query optimization.

56. What is join dependency and inclusion dependency?

Join Dependency:

A Join dependency is generalization of Multivalued dependency.A JD {R1, R2, ..., Rn} is said to hold over a relation R if R1, R2, R3, ..., Rn is a lossless-join decomposition of R . There is no set of sound and complete inference rules for JD.

Inclusion Dependency:

An Inclusion Dependency is a statement of the form that some columns of a relation are contained in other columns. A foreign key constraint is an example of inclusion dependency.

57. What is durability in DBMS?

Once the DBMS informs the user that a transaction has successfully completed, its effects should persist even if the system crashes before all its changes are reflected on disk. This property is called durability.

58. What do you mean by atomicity and aggregation?

Atomicity:

Either all actions are carried out or none are. Users should not have to worry about the effect of incomplete transactions. DBMS ensures this by undoing the actions of incomplete transactions.

Aggregation:

A concept which is used to model a relationship between a collection of entities and relationships. It is used when we need to express a relationship among relationships.

59. What is a Phantom Deadlock?

In distributed deadlock detection, the delay in propagating local information might cause the deadlock detection algorithms to identify deadlocks that do not really exist. Such situations are called phantom deadlocks and they lead to unnecessary aborts.

60. What is a checkpoint and When does it occur?

A Checkpoint is like a snapshot of the DBMS state. By taking checkpoints, the DBMS can reduce the amount of work to be done during restart in the event of subsequent crashes.

61. What are the different phases of transaction?

Different phases are

Analysis phase
Redo Phase
Undo phase

62. What do you mean by flat file database?

It is a database in which there are no programs or user access languages. It has no cross-file capabilities but is user-friendly and provides user-interface management.

63. What is "transparent DBMS"?

It is one, which keeps its Physical Structure hidden from user.

64. Brief theory of Network, Hierarchical schemas and their properties

Network schema uses a graph data structure to organize records example for such a database management system is CTCG while a hierarchical schema uses a tree data structure example for such a system is IMS.

65. What is a query?

A query with respect to DBMS relates to user commands that are used to interact with a data base. The query language can be classified into data definition language and data manipulation language.

66. What do you mean by Correlated subquery?

Subqueries, or nested queries, are used to bring back a set of rows to be used by the parent query. Depending on how the subquery is written, it can be executed once for the parent query or it can be executed once for each row returned by the parent query. If the subquery is executed for each row of the parent, this is called a *correlated subquery*.

A correlated subquery can be easily identified if it contains any references to the parent subquery columns in its WHERE clause. Columns from the subquery cannot be referenced anywhere else in the parent query. The following example demonstrates a non-correlated subquery.

E.g. Select * From CUST Where '10/03/1990' IN (Select ODATE From ORDER Where CUST.CNUM = ORDER.CNUM)

67. What are the primitive operations common to all record management systems?

Addition, deletion and modification.

68. Name the buffer in which all the commands that are typed in are stored

'Edit' Buffer

69. What are the unary operations in Relational Algebra?

PROJECTION and SELECTION.

70. Are the resulting relations of PRODUCT and JOIN operation the same?

No.
PRODUCT: Concatenation of every row in one relation with every row in another.
JOIN: Concatenation of rows from one relation and related rows from another.

71. What is RDBMS KERNEL?

Two important pieces of RDBMS architecture are the kernel, which is the software, and the data dictionary, which consists of the system-level data structures used by the kernel to manage the database

You might think of an RDBMS as an operating system (or set of subsystems), designed specifically for controlling data access; its primary functions are storing, retrieving, and securing data. An RDBMS maintains its own list of authorized users and their associated privileges; manages memory caches and paging; controls locking for concurrent resource usage; dispatches and schedules user requests; and manages space usage within its table-space structures

.

72. Name the sub-systems of a RDBMS

I/O, Security, Language Processing, Process Control, Storage Management, Logging and Recovery, Distribution Control, Transaction Control, Memory Management, Lock Management

73. Which part of the RDBMS takes care of the data dictionary? How

Data dictionary is a set of tables and database objects that is stored in a special area of the database and maintained exclusively by the kernel.

74. What is the job of the information stored in data-dictionary?

The information in the data dictionary validates the existence of the objects, provides access to them, and maps the actual physical storage location.

75. Not only RDBMS takes care of locating data it also

determines an optimal access path to store or retrieve the data

76. How do you communicate with an RDBMS?

You communicate with an RDBMS using Structured Query Language (SQL)

77. Define SQL and state the differences between SQL and other conventional programming Languages

SQL is a nonprocedural language that is designed specifically for data access operations on normalized relational database structures. The primary difference between SQL and other conventional programming languages is that SQL statements specify what data operations should be performed rather than how to perform them.

78. Name the three major set of files on disk that compose a database in Oracle

There are three major sets of files on disk that compose a database. All the files are binary. These are

Database files
Control files
Redo logs

The most important of these are the database files where the actual data resides. The control files and the redo logs support the functioning of the architecture itself.

All three sets of files must be present, open, and available to Oracle for any data on the database to be useable. Without these files, you cannot access the database, and the database administrator might have to recover some or all of the database using a backup, if there is one.

79. What is an Oracle Instance?

The Oracle system processes, also known as Oracle background processes, provide functions for the user processes—functions that would otherwise be done by the user processes themselves

Oracle database-wide system memory is known as the SGA, the *system global area* or *shared global area*. The data and control structures in the SGA are shareable, and all the Oracle background processes and user processes can use them.

The combination of the SGA and the Oracle background processes is known as an *Oracle instance*

80. What are the four Oracle system processes that must always be up and running for the database to be useable

The four Oracle system processes that must always be up and running for the database to be useable include *DBWR* (Database Writer), *LGWR* (Log Writer), *SMON* (System Monitor), and *PMON* (Process Monitor).

81. What are database files, control files and log files. How many of these files should a database have at least? Why?

Database Files

The database files hold the actual data and are typically the largest in size. Depending on their sizes, the tables (and other objects) for all the user accounts can go in one database file—but that's not an ideal situation because it does not make the database structure very flexible for controlling access to storage for different users, putting the database on different disk drives, or backing up and restoring just part of the database.

You must have at least one database file but usually, more than one files are used. In terms of accessing and using the data in the tables and other objects, the number (or location) of the files is immaterial.

The database files are fixed in size and never grow bigger than the size at which they were created

Control Files

The control files and redo logs support the rest of the architecture. Any database must have at least one control file, although you typically have more than one to guard against loss. The control file records the name of the database, the date and time it was created, the location of the database and redo logs, and the synchronization information to ensure that all three sets of files are always in step. Every time you add a new database or redo log file to the database, the information is recorded in the control files.

Redo Logs

Any database must have at least two redo logs. These are the journals for the database; the redo logs record all changes to the user objects or system objects. If any type of failure occurs, the changes recorded in the redo logs can be used to bring the database to a consistent state without losing any committed transactions. In the case of non-data loss failure, Oracle can apply the information in the redo logs automatically without intervention from the DBA.

The redo log files are fixed in size and never grow dynamically from the size at which they were created.

82. What is ROWID?

The ROWID is a unique database-wide physical address for every row on every table. Once assigned (when the row is first inserted into the database), it never changes until the row is deleted or the table is dropped.

The ROWID consists of the following three components, the combination of which

uniquely identifies the physical storage location of the row.

Oracle database file number, which contains the block with the rows
Oracle block address, which contains the row
The row within the block (because each block can hold many rows)

The ROWID is used internally in indexes as a quick means of retrieving rows with a particular key value. Application developers also use it in SQL statements as a quick way to access a row once they know the ROWID

83. What is Oracle Block? Can two Oracle Blocks have the same address?

Oracle "formats" the database files into a number of Oracle blocks when they are first created—making it easier for the RDBMS software to manage the files and easier to read data into the memory areas.

The block size should be a multiple of the operating system block size. Regardless of the block size, the entire block is not available for holding data; Oracle takes up some space to manage the contents of the block. This block header has a minimum size, but it can grow.

These Oracle blocks are the smallest unit of storage. Increasing the Oracle block size can improve performance, but it should be done only when the database is first created.

Each Oracle block is numbered sequentially for each database file starting at 1. Two blocks can have the same block address if they are in different database files.

84. What is database Trigger?

A database trigger is a PL/SQL block that can defined to automatically execute for insert, update, and delete statements against a table. The trigger can e defined to execute once for the entire statement or once for every row that is inserted, updated, or deleted. For any one table, there are twelve events for which you can define database triggers. A database trigger can call database procedures that are also written in PL/SQL.

85. Name two utilities that Oracle provides, which are use for backup and recovery.

Along with the RDBMS software, Oracle provides two utilities that you can use to back up and restore the database. These utilities are *Export* and *Import*.

The *Export utility* dumps the definitions and data for the specified part of the database to an operating system binary file. The *Import utility* reads the file produced by an export, recreates the definitions of objects, and inserts the data

If Export and Import are used as a means of backing up and recovering the database, all the changes made to the database cannot be recovered since the export was performed. The best you can do is recover the database to the time when the export was last performed.

86. What are stored-procedures? And what are the advantages of using them.

Stored procedures are database objects that perform a user defined operation. A stored procedure can have a set of compound SQL statements. A stored procedure executes the SQL commands and returns the result to the client. Stored procedures are used to reduce network traffic.

87. How are exceptions handled in PL/SQL? Give some of the internal exceptions' name

PL/SQL exception handling is a mechanism for dealing with run-time errors encountered during procedure execution. Use of this mechanism enables execution to continue if the error is not severe enough to cause procedure termination.

The exception handler must be defined within a subprogram specification. Errors cause the program to raise an exception with a transfer of control to the exception-handler block. After the exception handler executes, control returns to the block in which the handler was defined. If there are no more executable statements in the block, control returns to the caller.

User-Defined Exceptions

PL/SQL enables the user to define exception handlers in the declarations area of subprogram specifications. User accomplishes this by naming an exception as in the following example:

```
ot_failure   EXCEPTION;
```

In this case, the exception name is ot_failure. Code associated with this handler is written in the EXCEPTION specification area as follows:

```
EXCEPTION
    when OT_FAILURE then
      out_status_code := g_out_status_code;
      out_msg         := g_out_msg;
```

The following is an example of a subprogram exception:

```
EXCEPTION
    when NO_DATA_FOUND then
      g_out_status_code := 'FAIL';
      RAISE ot_failure;
```

Within this exception is the RAISE statement that transfers control back to the ot_failure exception handler. This technique of raising the exception is used to invoke all user-defined exceptions.

System-Defined Exceptions

Exceptions internal to PL/SQL are raised automatically upon error. NO_DATA_FOUND is a system-defined exception. Table below gives a complete list of internal exceptions.

PL/SQL internal exceptions.

Exception Name	***Oracle Error***
CURSOR_ALREADY_OPEN	ORA-06511
DUP_VAL_ON_INDEX	ORA-00001
INVALID_CURSOR	ORA-01001
INVALID_NUMBER	ORA-01722
LOGIN_DENIED	ORA-01017
NO_DATA_FOUND	ORA-01403
NOT_LOGGED_ON	ORA-01012
PROGRAM_ERROR	ORA-06501

STORAGE_ERROR	ORA-06500
TIMEOUT_ON_RESOURCE	ORA-00051
TOO_MANY_ROWS	ORA-01422
TRANSACTION_BACKED_OUT	ORA-00061
VALUE_ERROR	ORA-06502
ZERO_DIVIDE	ORA-01476

In addition to this list of exceptions, there is a catch-all exception named *OTHERS* that traps all errors for which specific error handling has not been established.

88. Does PL/SQL support "overloading"? Explain

The concept of *overloading* in PL/SQL relates to the idea that you can define procedures and functions with the same name. PL/SQL does not look only at the referenced name, however, to resolve a procedure or function call. The count and data types of formal parameters are also considered.

PL/SQL also attempts to resolve any procedure or function calls in locally defined packages before looking at globally defined packages or internal functions. To further ensure calling the proper procedure, you can use the dot notation. Prefacing a procedure or function name with the package name fully qualifies any procedure or function reference.

89. Tables derived from the ERD

a) Are totally unnormalised
b) Are always in 1NF
c) Can be further denormalised
d) May have multi-valued attributes

(b) Are always in 1NF

90. Spurious tuples may occur due to

i. Bad normalization
ii. Theta joins
iii. Updating tables from join

a) i & ii b) ii & iii
c) i & iii d) ii & iii

(a) i & iii because theta joins are joins made on keys that are not primary keys.

91. A B C is a set of attributes. The functional dependency is as follows

AB -> B
AC -> C
C -> B

a) is in 1NF
b) is in 2NF
c) is in 3NF
d) is in BCNF

(a) is in 1NF since $(AC)^+$ = { A, B, C} hence AC is the primary key. Since C B is a FD given, where neither C is a Key nor B is a prime attribute, this it is not in 3NF. Further B is not functionally dependent on key AC thus it is not in 2NF. Thus the given FDs is in 1NF.

92. In mapping of ERD to DFD

a) entities in ERD should correspond to an existing entity/store in DFD
b) entity in DFD is converted to attributes of an entity in ERD
c) relations in ERD has 1 to 1 correspondence to processes in DFD
d) relationships in ERD has 1 to 1 correspondence to flows in DFD

(a) entities in ERD should correspond to an existing entity/store in DFD

93. A dominant entity is the entity

a) on the N side in a 1 : N relationship
b) on the 1 side in a 1 : N relationship
c) on either side in a 1 : 1 relationship
d) nothing to do with 1 : 1 or 1 : N relationship

(b) on the 1 side in a 1 : N relationship

94. Select 'NORTH', CUSTOMER From CUST_DTLS Where REGION = 'N' Order By CUSTOMER Union Select 'EAST', CUSTOMER From CUST_DTLS Where REGION = 'E' Order By CUSTOMER
The above is

a) Not an error
b) Error - the string in single quotes 'NORTH' and 'SOUTH'
c) Error - the string should be in double quotes
d) Error - ORDER BY clause

(d) Error - the ORDER BY clause. Since ORDER BY clause cannot be used in UNIONS

95. What is Storage Manager?

It is a program module that provides the interface between the low-level data stored in database, application programs and queries submitted to the system.

96. What is Buffer Manager?

It is a program module, which is responsible for fetching data from disk storage into main memory and deciding what data to be cache in memory.

97. What is Transaction Manager?

It is a program module, which ensures that database, remains in a consistent state despite system failures and concurrent transaction execution proceeds without conflicting.

98. What is File Manager?

It is a program module, which manages the allocation of space on disk storage and data structure used to represent information stored on a disk.

99. What is Authorization and Integrity manager?

It is the program module, which tests for the satisfaction of integrity constraint and checks the authority of user to access data.

100. What are stand-alone procedures?

Procedures that are not part of a package are known as stand-alone because they independently defined. A good example of a stand-alone procedure is one written in a SQL*Forms application. These types of procedures are not available for reference from other Oracle tools. Another limitation of stand-alone procedures is that they are compiled at run time, which slows execution.

101. What are cursors give different types of cursors.

PL/SQL uses cursors for all database information accesses statements. The language supports the use two types of cursors

Implicit
Explicit

102. What is cold backup and hot backup (in case of Oracle)?

Cold Backup:

It is copying the three sets of files (database files, redo logs, and control file) when the instance is shut down. This is a straight file copy, usually from the disk directly to tape. You must shut down the instance to guarantee a consistent copy.

If a cold backup is performed, the only option available in the event of data file loss is restoring all the files from the latest backup. All work performed on the database since the last backup is lost.

Hot Backup:

Some sites (such as worldwide airline reservations systems) cannot shut down the database while making a backup copy of the files. The cold backup is not an available option.

So different means of backing up database must be used — the hot backup. Issue a SQL command to indicate to Oracle, on a tablespace-by-tablespace basis, that the files of the tablespace are to backed up. The users can continue to make full use of the files, including making changes to the data. Once the user has indicated that he/she wants to back up the tablespace files, he/she can use the operating system to copy those files to the desired backup

destination.

The database must be running in ARCHIVELOG mode for the hot backup option.

If a data loss failure does occur, the lost database files can be restored using the hot backup and the online and offline redo logs created since the backup was done. The database is restored to the most consistent state without any loss of committed transactions.

103. What are Armstrong rules? How do we say that they are complete and/or sound

The well-known inference rules for FDs

Reflexive rule :

If Y is subset or equal to X then X Y.

Augmentation rule:

If X Y then XZ YZ.

Transitive rule:

If {X Y, Y Z} then X Z.

Decomposition rule :

If X YZ then X Y.

Union or Additive rule:

If {X Y, X Z} then X YZ.

Pseudo Transitive rule :

If {X Y, WY Z} then WX Z.

Of these the first three are known as Amstrong Rules. They are sound because it is enough if a set of FDs satisfy these three. They are called complete because using these three rules we can generate the rest all inference rules.

104. How can you find the minimal key of relational schema?

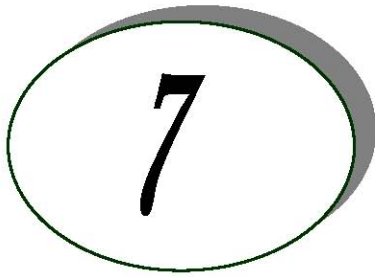
Minimal key is one which can identify each tuple of the given relation schema uniquely. For finding the minimal key it is required to find the closure that is the set of all attributes that are dependent on any given set of attributes under the given set of functional dependency.

Algo. I Determining X^+, closure for X, given set of FDs F

1. Set $X^+ = X$

Set Old $X^+ = X^+$

FD $Y \rightarrow Z$



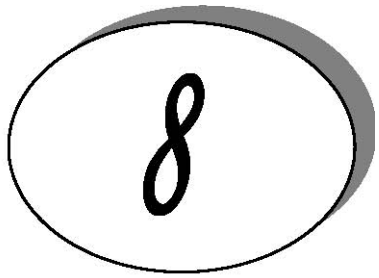
SQL

The query checks whether a given string is a numerical digit.

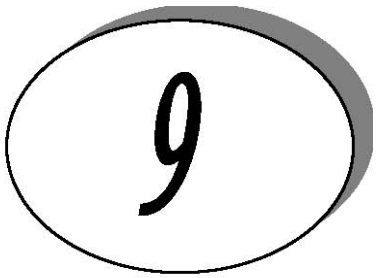
19. What does the following query do?

SELECT SAL + NVL(COMM,0) FROM EMP;

This displays the total salary of all employees. The null values in the commission column will be replaced by 0 and added to salary.



Computer Networks



Operating Systems