SEVENTH SENSE
Talent Solutions

Technical Aptitude Questions

1. 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;
```

```
        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 to each other.	Provides response to a message.
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.

3.Data Structures

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.

1. *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

2. *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

3. *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.

4. *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.

5. *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.

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

Polish and Reverse Polish notations.

7. *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 + ^

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

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

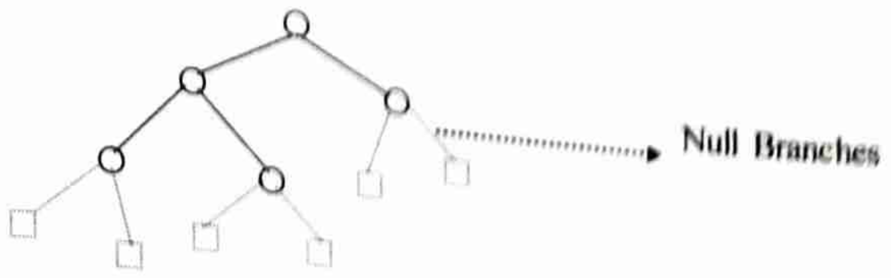
(d) Deletion.

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.

9. *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.*

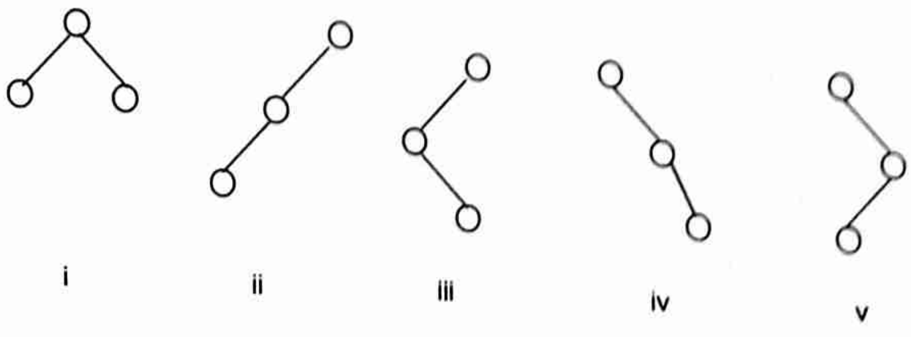
10. *What are the methods available in storing sequential files ?*

- Straight merging,
- Natural merging,
- Polyphase sort,
- Distribution of Initial runs.

11. *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.



In general:

*If there are **n** nodes, there exist **2^n-n** different trees.*

12. *List out few of the Application of tree data-structure?*

- The manipulation of Arithmetic expression,

- Symbol Table construction,
- Syntax analysis.

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

- Sparse matrix,
- Index generation.

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

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

(b) Linked list

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

Backtracking

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

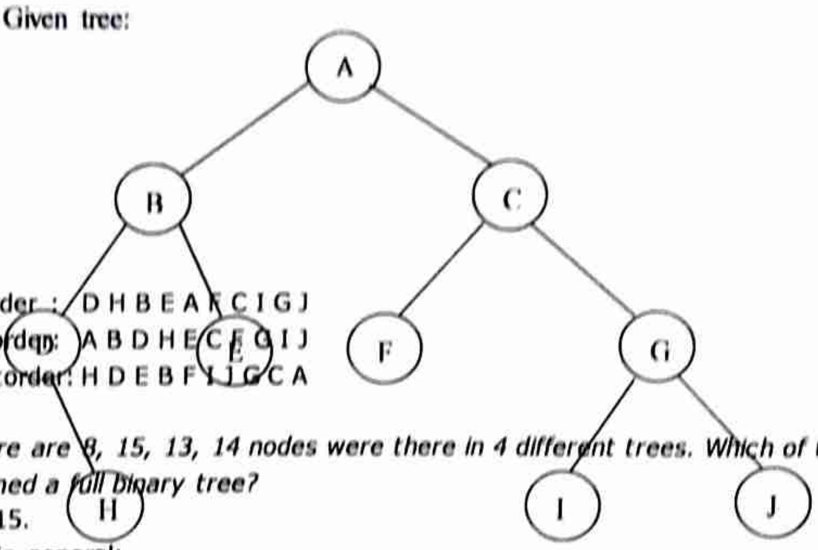
If the 'pivotal value' (or the 'Height factor') is greater than 1 or less than -1.

17. 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.

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

Given tree:



- 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

19. 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:

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

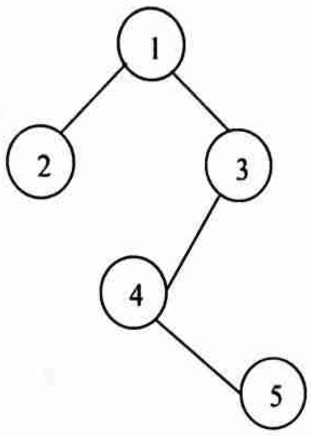
By the method of elimination:

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.*

20. 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

21. 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

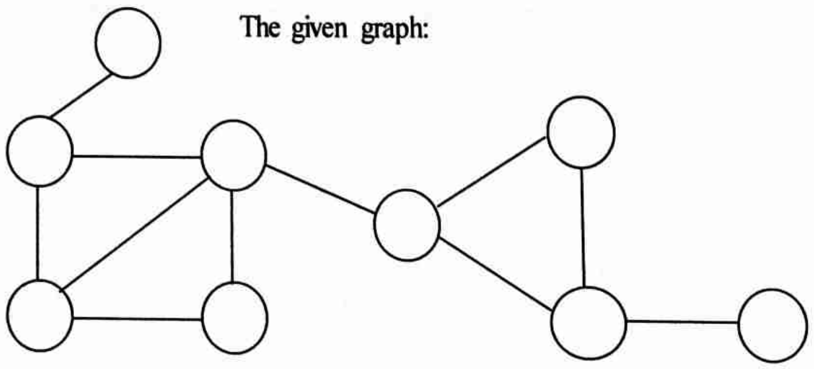
When the L and R pointers cross each other the pivot value is interchanged with the value at right pointer. If the pivot is changed it means that the pivot has occupied its original position in the sorted order (shown in bold italics) and hence two different arrays are formed, one from start of the original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

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



- *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.

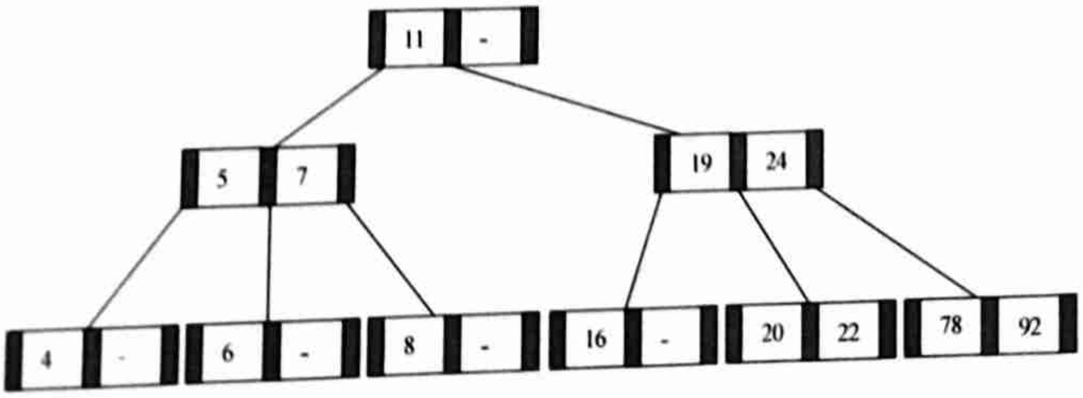
25. What are the types of Collision Resolution Techniques and the methods used in each of the type?

- 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?

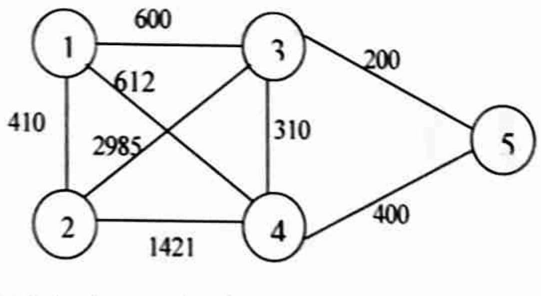
A spanning tree is a tree associated with a network. All the nodes of the graph appear on the tree once. A minimum spanning tree is a spanning tree organized so that the total edge weight between nodes is minimized.

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

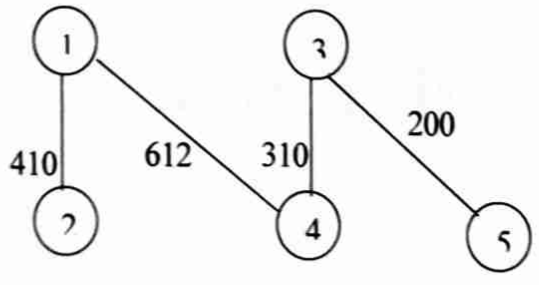
No.

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?*

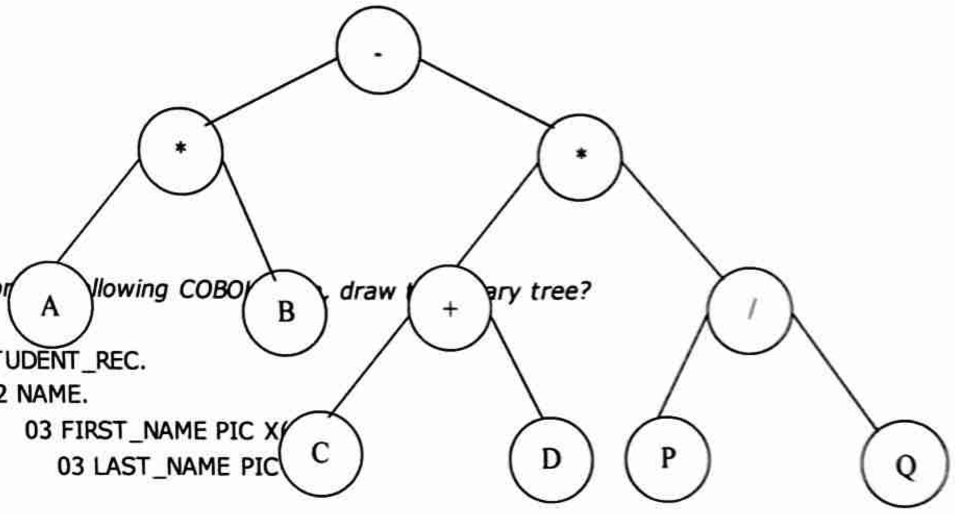
(a) Sequential
(b) Indexed
(c) Random

(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 :*

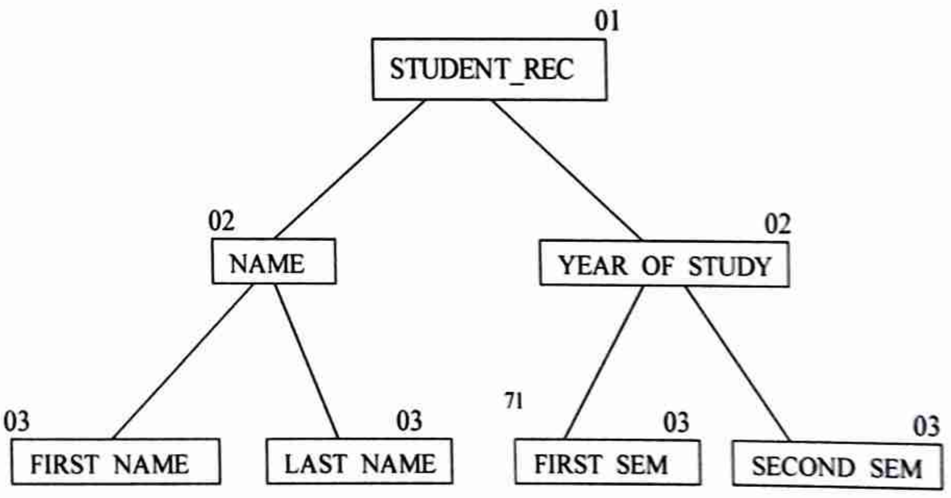
*A * B - (C + D) * (P / Q)*



35. *Fo llowing COBO , draw ary tree?*

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

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



01
STUDENT_REC
02
NAME
02
YEAR OF STUDY
03
FIRST NAME
03
LAST NAME
03
FIRST SEM
03
SECOND SEM