

EC33 DATA STRUCTURES and OBJECT ORIENTED PROGRAMMING

UNIT I

1. State the characteristics of procedure oriented programming.

- Emphasis is on algorithm.
- Large programs are divided into smaller programs called functions.
- Functions share global data.
- Data move openly around the system from function to function.
- Functions transform data from one form to another.
- Employs top-down approach in program design.

2. What are the features of Object Oriented Programming?

- Emphasis is on data rather than procedure.
- Programs are divided into objects.
- Data structures are designed such that they characterize the objects.
- Functions that operate on the data of an object are tied together.
- Data is hidden and cannot be accessed by external functions.
- Objects may communicate with each other through functions.
- New data and functions can easily be added whenever necessary.
- Follows bottom-up approach.

3. Distinguish between Procedure Oriented Programming and Object Oriented Programming.

- Procedure Oriented Programming
- Object Oriented Programming
- Emphasis is on algorithm.
- Large programs are divided into smaller programs called functions.
- Functions share global data.
- Data move openly around the system from function to function.
- Employs top-down approach in program design.
- Emphasis is on data rather than procedure.
- Programs are divided into objects.
- Functions that operate on the data of an object are tied together.
- Data is hidden and cannot be accessed by external functions.
- Follows bottom-up approach.

4. Define Object Oriented Programming (OOP).

Object Oriented Programming is an approach that provides a way of modularizing programs by creating partitioned memory area for both data and functions that can be used as templates for creating copies of such modules on demand.

5. List out the basic concepts of Object Oriented Programming.

- Objects
- Classes
- Data Abstraction and Encapsulation
- Inheritance
- Polymorphism
- Dynamic Binding
- Message Passing

6. Define Objects.

Objects are the basic run time entities in an object oriented system. They are instance of a class. They may represent a person, a place etc that a program has to handle. They may also represent user-defined data. They contain both data and code.

7. Define Class.

Class is a collection of objects of similar data types. Class is a user-defined data type. The entire set of data and code of an object can be made a user defined type through a class.

8. Define Encapsulation and Data Hiding.

The wrapping up of data and functions into a single unit is known as data encapsulation. Here the data is not accessible to the outside world. The insulation of data from direct access by the program is called data hiding or information hiding.

9. Define Data Abstraction.

Abstraction refers to the act of representing the essential features without including the background details or explanations.

10. Define data members and member functions.

The attributes in the objects are known as data members because they hold the information. The functions that operate on these data are known as methods or member functions.

11. State Inheritance.

Inheritance is the process by which objects of one class acquire the properties of objects of another class. It supports the concept of hierarchical classification and provides the idea of reusability. The class which is inherited is known as the base or super class and class which is newly derived is known as the derived or sub class.

12. State Polymorphism.

Polymorphism is an important concept of OOPs. Polymorphism means one name, multiple forms. It is the ability of a function or operator to take more than one form at different instances.

13. List and define the two types of Polymorphism.

- Operator Overloading – The process of making an operator to exhibit different behaviors at different instances.
- Function Overloading – Using a single function name to perform different types of tasks. The same function name can be used to handle different number and different types of arguments.

14. State Dynamic Binding.

Binding refers to the linking of procedure call to the code to be executed in response to the call. Dynamic Binding or Late Binding means that the code associated with a given procedure call is known only at the run-time.

15. Define Message Passing.

Objects communicate between each other by sending and receiving information known as messages. A message to an object is a request for execution of a procedure. Message passing involves specifying the name of the object, the name of the function and the information to be sent.

16. List out some of the benefits of OOP.

- Eliminate redundant code
- Saves development time and leads to higher productivity
- Helps to build secure programs
- Easy to partition work
- Small programs can be easily upgraded to large programs
- Software complexity can easily be managed

17. Define Object Based Programming language.

Object Based Programming is the style of programming that primarily supports encapsulation and object identity. Languages that support programming with objects are known as Object Based Programming languages. They do not support inheritance and dynamic binding.

18. List out the applications of OOP.

- Real time systems
- Simulation and modeling
- Object oriented databases
- Hypertext, Hypermedia and expertext
- AI and expert systems
- Neural networks and parallel programming
- Decision support and office automation systems
- CIM/CAM/CAD systems

19. Define C++.

C++ is an object oriented programming language developed by Bjarne Stroustrup. It is a super set of C. Initially it was known as "C with Classes". It is a versatile language for handling large programs.

20. What are the input and output operators used in C++?

The identifier cin is used for input operation. The input operator used is >>, which is known as the extraction or get from operator. The syntax is, cin >> n1;
The identifier cout is used for output operation. The input operator used is <<, which is known as the insertion or put to operator. The syntax is, cout << "C++ is better than C";

21. What is the return type of main ()?

The return type of main () is integer i.e. main () returns an integer type value to the operating system. Therefore, every main () should end with a return (0) statement. It's general format is,

```
int main ()  
{  
.....  
return 0;  
}
```

22. List out the four basic sections in a typical C++ program.

- Include files
- Class declaration
- Member functions definition
- Main function program

23. Define token. What are the tokens used in C++?

The smallest individual units in a program are known as tokens. The various tokens in C++ are keywords, identifiers, constants, strings and operators.

24. Define identifier. What are the rules to be followed for identifiers?

Identifiers refer to the names of variables, functions, arrays, classes etc created by the programmer. The rules are as follows:

- Only alphabetic characters, digits and underscores are permitted
- The name cannot start with a digit
- Uppercase and lowercase letters are distinct
- A declared keyword cannot be used as a variable name

25. State the use of void in C++.

The two normal uses of void are

- To specify the return type of the function when it is not returning a value
- To indicate an empty argument list to a function

26. Define an enumeration data type.

An enumerated data type is a user defined data type that provides a way for attaching names to numbers thereby increasing comprehensibility of the code. The enum keyword is used which automatically enumerates a list of words by assigning them values 0, 1, 2...

E.g.) enum shape {circle, square, triangle};

27. Define constant pointer and pointer to a constant.

The constant pointer concept does not allow us to modify the value initialized to the pointer.

e.g.) char * const ptr = "GOOD";

The pointer to a constant concept doesn't allow us to modify the address of the pointer.

E.g.) int const * ptr = &n;

28. What are the two ways of creating symbolic constants?

- Using the qualifier const
- Defining a set of integer constants using enum keyword

29. Define reference variable. Give its syntax.

A reference variable provides an alias or alternate name for a previously defined variable. It must be initialized at the time of declaration. Its syntax is given by, data-type & reference-name = variable-name;

30. List out the new operators introduced in C++.

- :: Scope resolution operator
- ::* Pointer to member declarator
- ->* Pointer to member operator
- .* Pointer to member operator
- delete Memory release operator
- endl Line feed operator
- new Memory allocation operator
- setw Field width operator

31. What is the use of scope resolution operator?

A variable declared in an inner block cannot be accessed outside the block. To resolve this problem the scope resolution operator is used. It can be used to uncover a hidden variable. This operator allows access to the global version of the variable. It takes the form, :: variable-name

32. List out the memory differencing operator.

- ::* To declare a pointer to the member of the class
- ->* To access a member using object name and a pointer to that member
- .* To access a member using a pointer to the object and a pointer to that member

33. Define the 2 memory management operators.

- new Memory allocation operator

The new operator can be used to create objects of any data-type. It allocates sufficient memory to hold a data object of type data-type and returns the address of the object.

Its general form is, Pointer variable=new data-type;

- delete Memory release operator

When a data object is no longer needed it is destroyed to release the memory space for reuse. The general form is, delete pointer variable;

34. List out the advantages of new operator over malloc ().

It automatically computes the size of the data object.

It automatically returns the correct pointer type.

It is possible to initialize the objects while creating the memory space.

It can be overloaded.

35. Define manipulators. What are the manipulators used in C++?

Manipulators are operators that are used to format the data display. The manipulators used in C++ are

endl – causes a linefeed to be inserted

setw – provides a common field width for all the numbers and forces them to be printed right justified

36. What are the three types of special assignment expressions?

Chained assignment e.g., x = y = 10;

Embedded assignment e.g., x = (y = 50) + 10;

Compound assignment e.g., x += 10;

37. Define implicit conversion.

Whenever data types are mixed in a expression, C++ performs the conversions automatically. This process is known as implicit or automatic conversion.

e.g., m = 5 + 2.75;

38. Define integral widening conversion.

Whenever a char or short int appears in an expression, it is converted to an int. This is called integral widening conversion.

39. What are the control structures used in C++?

- Sequence structure (straight line)
- Selection structure (branching)
 - _ if – else (two way branch)
 - _ switch (multiple branch)
- Loop structure (iteration or repetition)
 - _ do – while (exit controlled)
 - _ while (entry controlled)
 - _ for (entry controlled)

40. Define Function Prototyping.

The function prototype describes the function interface to the compiler by giving details such as the number and type of arguments and type of return values. It is the declaration of a function in a program. It is in the following form, type function – name (argument – list);
where argument – list -> types and names of arguments to be passed to the function

41. What is call by reference?

When we pass arguments by reference, the formal arguments in the called function become the aliases to the actual arguments in the calling function. Here the function works on the original data rather than its copy.

e.g., void swap (int &a, int &b)

```
{  
int t = a;  
a = b;  
b = t;  
}
```

42. What are inline functions?

An inline function is a function that is expanded in line when it is invoked. Here, the compiler replaces the function call with the corresponding function code. The inline function is defined as,
inline function-header

```
{  
function body  
}
```

43. List out the conditions where inline expansion doesn't work.

For functions returning values, if a loop, a switch, or a goto exists

For functions not returning values, if a return statement exists

If functions contain static variables

If inline functions are recursive

44. Why do we use default arguments?

The function assigns a default value to the parameter which does not have a matching argument in the function call. They are useful in situations where some arguments always have the same value.

e.g., float amt (float P, float n, float r = 0.15);

45. State the advantages of default arguments.

The advantages of default arguments are,

We can use default arguments to add new parameters to the existing function.

Default arguments can be used to combine similar functions into one.

46. Define function overloading.

A single function name can be used to perform different types of tasks. The same function name can be used to handle different number and different types of arguments. This is known as function overloading or function polymorphism.

47. List out the limitations of function overloading.

We should not overload unrelated functions and should reserve function overloading for functions that perform closely related operations.

UNIT II

1. State the difference between structures and class.

By default the members of a structure are public whereas the members of a class are private.

2. Define a class.

A class is a way to bind the data and its function together. It allows the data to be hidden from external use. The general form of a class is,

```
class class_name
{
private:
variable declarations;
function declaration;
public:
variable declarations;
function declaration;
};
```

3. List the access modes used within a class.

Private – The class members are private by default. The members declared private are completely hidden from the outside world. They can be accessed from only within the class.

Public – The class members declared public can be accessed from any where.

Protected – The class members declared protected can be access from within the class and also by the friend classes.

4. How can we access the class members?

The class members can be accessed only when an object is created to that class. They are accessed with the help of the object name and a dot operator.

They can be accessed using the general format,
Object_name.function_name (actual_arguments);

5. Where can we define member functions?

Member functions can be defined in two places:

Outside the class definition – The member functions can be defined outside the class definition with the help of the scope resolution operator.

The general format is given as,

```
return_type class_name :: function_name (argument declaration)
{
function body
}
```

Inside the class definition – The member function is written inside the class in place of the member declaration. They are treated as inline functions.

6. What are the characteristics of member functions?

The various characteristics of member functions are,

Different classes can use the same function name and their scope can be resolved using the membership label.

Member functions can access the private data of a class while a nonmember function cannot.

A member function can call another member function directly without using a dot operator.

7. How can an outside function be made inline?

An outside function can be made inline by just using the qualifier 'inline' in the header line of the function definition.

The general format is,

```
inline return_type class_name :: function_name (argument declaration)
{
function body
}
```

8. What are the properties of a static data member?

The properties of a static data member are,

It is initialized to zero when the first object is created and no other initialization is permitted.

Only one copy of that member is created and shared by all the objects of that class.

It is visible only within the class, but the life time is the entire program.

9. What are the properties of a static member function?

A static member function can access only other static members declared in the same class.

It can be called using the class name instead of objects as follows,
class_name :: function_name;

10. How can objects be used as function arguments?

An object can be used as a function argument in two ways,

A copy of the entire object is passed to the function. (Pass by value)

Only the address of the object is transferred to the function. (Pass by reference)

11. Define friend function?

An outside function can be made a friend to a class using the qualifier 'friend'. The function declaration should be preceded by the keyword friend. A friend function has full access rights to the private members of a class.

12. List out the special characteristics of a friend function.

- It is not in the scope of a class in which it is declared as friend.
- It cannot be called using the object of that class.
- It can be invoked without an object.
- It cannot access the member names directly and uses the dot operator.
- It can be declared as either public or private.
- It has the objects as arguments.

13. Define Constructor.

A constructor is a special member function whose task is to initialize the objects of its class. It has the same name as the class. It gets invoked whenever an object is created to that class. It is called so since it constructs the values of data members of the class.

14. List some of the special characteristics of constructor.

- Constructors should be declared in the public section.
- They are invoked automatically when the objects are created.
- They do not have return types
- They cannot be inherited.

15. Give the various types of constructors.

There are four types of constructors. They are

- Default constructors – A constructor that accepts no parameters
- Parameterized constructors – The constructors that can take arguments
- Copy constructor – It takes a reference to an object of the same class as itself as an argument
- Dynamic constructors – Used to allocate memory while creating objects

16. What are the ways in which a constructor can be called?

The constructor can be called by two ways. They are,

- By calling the constructor explicitly
e.g., integer int1 = integer (0, 100);
- By calling the constructor implicitly
.g., integer int1 (0, 100);

17. State dynamic initialization of objects.

Class objects can be initialized dynamically. The initial values of an object may be provided during run time. The advantage of dynamic initialization is that various initialization formats can be used. It provides flexibility of using different data formats.

18. Define Destructor.

A destructor is used to destroy the objects that have been created by a constructor. It is a special member function whose name is same as the class and is preceded by a tilde '~' symbol.

19. Give the general form of an operator function.

The general form of an operator function is given as,
return-type class-name :: operator op (arglist)

```
{  
function body  
}
```

Where,

Return-type -> type of value returned

operator -> keyword

op -> operator being overloaded

20. List some of the rules for operator overloading.

Only existing operators can be overloaded.

We cannot change the basic meaning of an operator.

The overloaded operator must have at least one operand.

Overloaded operators follow the syntax rules of the original operators.

21. What are the types of type conversions?

There are three types of conversions. They are

Conversion from basic type to class type – done using constructor

Conversion from class type to basic type – done using a casting operator

Conversion from one class type to another – done using constructor or casting operator

22. What are the conditions should a casting operator satisfy?

The conditions that a casting operator should satisfy are,

It must be a class member.

It must not specify a return type.

It must not have any arguments.

23. Define implicit conversion.

Whenever data types are mixed in a expression, C++ performs the conversions automatically. This process is known as implicit or automatic conversion.

e.g., m = 5 + 2.75;

24. Define integral widening conversion.

Whenever a char or short int appears in an expression, it is converted to an int. This is called integral widening conversion.

25. What are the control structures used in C++?

- Sequence structure (straight line)
- Selection structure (branching)
 - _ if – else (two way branch)
 - _ switch (multiple branch)
- Loop structure (iteration or repetition)
 - _ do – while (exit controlled)
 - _ while (entry controlled)
 - _ for (entry controlled)

26. What are the types of inheritance?

The various types of inheritance are,

- Single inheritance
- Multi-level inheritance
- Multiple inheritance
- Hierarchical inheritance
- Hybrid inheritance

27. Give the syntax for inheritance.

The syntax of deriving a new class from an already existing class is given by,

```
class derived-class : visibility-mode base-class
{
body of derived class
}
```

28. Define single inheritance.

In single inheritance, one class is derived from an already existing base class. Here A is the base class and B is the derived class.

29. Define multi-level inheritance.

In multi-level inheritance, a new class is derived from a class already derived from the base class. Here, class B is derived from class A and class C is further derived from the derived class B.

30. Define multiple inheritance.

In multiple inheritance, a single class is derived from more than one base class.

```
A
B
A
B
C
```

Here class C is derived from two base classes A and B.

31. Define Hierarchical inheritance.

In hierarchical inheritance, more than one class is derived from a single base class. Here class B and C are derived from class A.

32. Define Hybrid inheritance.

Hybrid inheritance is defined as a combination of more than one inheritance. Here, Classes A, B and C represent hierarchical inheritance. Classes A, B & D and classes A, C and D represent multilevel inheritance. Classes B, C and D represent multiple inheritance.

33. What is a virtual base class?

Here, class D inherits both classes B and C which are derived from the same base class A. Hence D has two copies of the properties of class A. This can be avoided by declaring classes B and C as virtual base classes.

A B
C
A
B C
A
B C
D
A
B C
D

34. What is an abstract class?

An abstract class is one that is not used to create objects. It is designed only to act as a base class to be inherited by other classes.

35. What are the types of polymorphism?

The two types of polymorphism are,

- Compile time polymorphism – The compiler selects the appropriate function for a particular call at the compile time itself. It can be achieved by function overloading and operator overloading.
- Run time Polymorphism - The compiler selects the appropriate function for a particular call at the run time only. It can be achieved using virtual functions.

36. Define 'this' pointer.

A 'this' pointer refers to an object that currently invokes a member function. For e.g., the function call a.show() will set the pointer 'this' to the address of the object 'a'.

37. What is a virtual function?

When a function is declared as virtual, C++ determines which function to use at run time based on the type of object pointed to by the base pointer, rather than the type of the pointer.

38. What is a pure virtual function?

A virtual function, equated to zero is called a pure virtual function. It is a function declared in a base class that has no definition relative to the base class.

39. How can a private member be made inheritable?

A private member can be made inheritable by declaring the data members as protected. When declared as protected the data members can be inherited by the friend classes.

40. Define implicit conversion.

Whenever data types are mixed in an expression, C++ performs the conversions automatically. This process is known as implicit or automatic conversion.

e.g., `m = 5 + 2.75;`

41. Define integral widening conversion.

Whenever a char or short int appears in an expression, it is converted to an int. This is called integral widening conversion.

42. What are the control structures used in C++?

- Sequence structure (straight line)
- Selection structure (branching)
 - _ if – else (two way branch)
 - _ switch (multiple branch)
- Loop structure (iteration or repetition)
 - _ do – while (exit controlled)
 - _ while (entry controlled)
 - _ for (entry controlled)

43. Define Function Prototyping.

The function prototype describes the function interface to the compiler by giving details such as the number and type of arguments and type of return values. It is the declaration of a function in a program. It is in the following form, type function – name (argument – list);

where argument – list -> types and names of arguments to be passed to the function

44. What is call by reference?

When we pass arguments by reference, the formal arguments in the called function become the aliases to the actual arguments in the calling function. Here the function works on the original data rather than its copy.

e.g., `void swap (int &a, int &b)`

```
{  
int t = a;  
a = b;  
b = t;  
}
```

45)What is boolean data type?

Java has simple type called boolean for logical values. It can have only one of two possible values, true or false. This is the type returned by all relational operators like

`a<b`.boolean is also required by the conditional expression that governs the control statements such as if for.

Student

Test

Result

Sports

Syntax: `boolean variablename;`

UNIT III

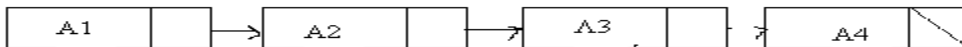
1. Define Data Structures

Data Structures is defined as the way of organizing all data items that consider not

only the elements stored but also stores the relationship between the elements.

2. Define Linked Lists

Linked list consists of a series of structures, which are not necessarily adjacent in memory. Each structure contains the element and a pointer to a structure containing its successor. We call this the Next Pointer. The last cell's Next pointer points to NULL.



3. State the different types of linked lists

The different types of linked list include singly linked list, doubly linked list and circular linked list.

4. List the basic operations carried out in a linked list

The basic operations carried out in a linked list include:

- Creation of a list
- Insertion of a node
- Deletion of a node
- Modification of a node
- Traversal of the list

5. List out the advantages of using a linked list

- It is not necessary to specify the number of elements in a linked list during its declaration
- Linked list can grow and shrink in size depending upon the insertion and deletion that occurs in the list

- Insertions and deletions at any place in a list can be handled easily and efficiently
- A linked list does not waste any memory space

6. List out the disadvantages of using a linked list

- Searching a particular element in a list is difficult and time consuming
- A linked list will use more storage space than an array to store the same number of elements

7. List out the applications of a linked list

Some of the important applications of linked lists are manipulation of polynomials, sparse matrices, stacks and queues.

8. State the difference between arrays and linked lists

Arrays	Linked Lists
Size of an array is fixed	Size of a list is variable
It is necessary to specify the number of elements during declaration.	It is not necessary to specify the number of elements during declaration
Insertions and deletions are somewhat difficult	Insertions and deletions are carried
It occupies less memory than a linked list for the same number of elements	It occupies more memory

9. Define a stack

Stack is an ordered collection of elements in which insertions and deletions are restricted to one end. The end from which elements are added and/or removed is referred to as top of the stack. Stacks are also referred as piles, push-down lists and last-in-first-out (LIFO) lists.

10. List out the basic operations that can be performed on a stack

The basic operations that can be performed on a stack are

- Push operation

- Pop operation
- Peek operation
- Empty check
- Fully occupied check

11. State the different ways of representing expressions

The different ways of representing expressions are

- Infix Notation
- Prefix Notation
- Postfix Notation

12. State the rules to be followed during infix to postfix conversions

- Fully parenthesize the expression starting from left to right. During parenthesizing, the operators having higher precedence are first parenthesized
- Move the operators one by one to their right, such that each operator replaces their corresponding right parenthesis
- The part of the expression, which has been converted into postfix is to be treated as single operand

13. State the difference between stacks and linked lists

The difference between stacks and linked lists is that insertions and deletions may

occur anywhere in a linked list, but only at the top of the stack

14. Mention the advantages of representing stacks using linked lists than arrays

- It is not necessary to specify the number of elements to be stored in a stack during its declaration, since memory is allocated dynamically at run time when an element is added to the stack
- Insertions and deletions can be handled easily and efficiently
- Linked list representation of stacks can grow and shrink in size without wasting memory space, depending upon the insertion and deletion that occurs in the list
- Multiple stacks can be represented efficiently using a chain for each stack

15. Define a queue

Queue is an ordered collection of elements in which insertions are restricted to one end called the rear end and deletions are restricted to other end called the front end. Queues are also referred as First-In-First-Out (FIFO) Lists.

16. Define a priority queue

Priority queue is a collection of elements, each containing a key referred as the priority for that element. Elements can be inserted in any order (i.e., of alternating priority), but are arranged in order of their priority value in the queue. The elements are deleted from the queue in the order of their priority (i.e., the elements with the highest priority is deleted first). The elements with the same priority are given equal importance and processed accordingly.

17. State the difference between queues and linked lists

The difference between queues and linked lists is that insertions and deletions may occur anywhere in the linked list, but in queues insertions can be made only in the rear end and deletions can be made only in the front end.

18. Define a Deque

Deque (Double-Ended Queue) is another form of a queue in which insertions and deletions are made at both the front and rear ends of the queue. There are two variations of a deque, namely, input restricted deque and output restricted deque. The input

restricted deque allows insertion at one end (it can be either front or rear) only. The

output restricted deque allows deletion at one end (it can be either front or rear) only.

19. Define an Abstract Data Type (ADT)

An abstract data type is a set of operations. ADTs are mathematical abstractions; nowhere in an ADT's definition is there any mention of how the set of operations is implemented. Objects such as lists, sets and graphs, along with their operations can be viewed as abstract data types.

20. What are the advantages of modularity?

- It is much easier to debug small routines than large routines
- It is easier for several people to work on a modular program simultaneously
- A well-written modular program places certain dependencies in only one routine, making changes easier

21. What are the objectives of studying data structures?

- To identify and create useful mathematical entities and operations to determine what classes of problems can be solved using these entities and operations

- To determine the representation of these abstract entities and to implement the abstract operations on these concrete representation

22. What are the types of queues?

- Linear Queues – The queue has two ends, the front end and the rear end. The rear end is where we insert elements and front end is where we delete elements. We can traverse in a linear queue in only one direction ie) from front to rear.

- Circular Queues – Another form of linear queue in which the last position is connected to the first position of the list. The circular queue is similar to linear queue has two ends, the front end and the rear end. The rear end is where we insert elements and front end is where we delete elements. We can traverse in a circular queue in only one direction ie) from front to rear.

- Double-Ended-Queue – Another form of queue in which insertions and deletions are made at both the front and rear ends of the queue.

23. List the applications of stacks

- Towers of Hanoi
- Reversing a string
- Balanced parenthesis
- Recursion using stack
- Evaluation of arithmetic expressions

24. List the applications of queues

- Jobs submitted to printer
- Real life line
- Calls to large companies
- Access to limited resources in Universities
- Accessing files from file server

25. Why we need cursor implementation of linked lists?

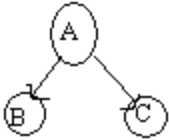
Many languages such as BASIC and FORTRAN do not support pointers. If linked lists are required and pointers are not available, then an alternative implementation must be used known as cursor implementation.

26. Define a tree

A tree is a collection of nodes. The collection can be empty; otherwise, a tree consists of a distinguished node r , called the root, and zero or more nonempty (sub) trees T_1, T_2, \dots, T_k , each of whose roots are connected by a directed edge from r .

27. Define root

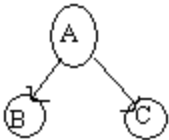
This is the unique node in the tree to which further sub-trees are attached.



Here, A is the root.

28. Define degree of the node

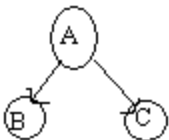
The total number of sub-trees attached to that node is called the degree of the node.



For node A, the degree is 2 and for B and C, the degree is 0.

29. Define leaves

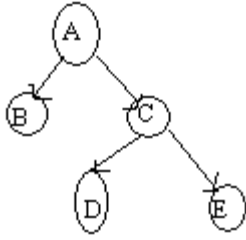
These are the terminal nodes of the tree. The nodes with degree 0 are always the leaves.



Here, B and C are leaf nodes.

30. Define internal nodes

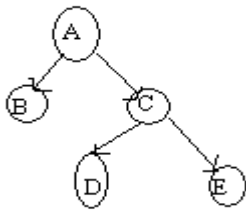
The nodes other than the root and the leaves are called internal nodes.



Here, C is the internal node.

31. Define parent node

The node which is having further sub-branches is called the parent node of those sub-branches.



Here, node C is the parent node of D and E

32. Define depth and height of a node

For any node n_i , the depth of n_i is the length of the unique path from the root to n_i . The height of n_i is the length of the longest path from n_i to a leaf.

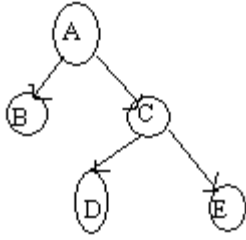
33. Define depth and height of a tree

The depth of the tree is the depth of the deepest leaf. The height of the tree is equal to the height of the root. Always depth of the tree is equal to height of the tree.

34. What do you mean by level of the tree?

The root node is always considered at level zero, then its adjacent children are supposed to be at level 1 and so on.

Here, node A is at level 0, nodes B and C are at level 1 and nodes D and E are at level 2.



35. Define forest

A tree may be defined as a forest in which only a single node (root) has no predecessors. Any forest consists of a collection of trees.

36. Define a binary tree

A binary tree is a finite set of nodes which is either empty or consists of a root and two disjoint binary trees called the left sub-tree and right sub-tree.

37. Define a path in a tree

A path in a tree is a sequence of distinct nodes in which successive nodes are connected by edges in the tree.

38. Define a full binary tree

A full binary tree is a tree in which all the leaves are on the same level and every non-leaf node has exactly two children.

39. Define a complete binary tree

A complete binary tree is a tree in which every non-leaf node has exactly two children not necessarily to be on the same level.

40. State the properties of a binary tree

- The maximum number of nodes on level n of a binary tree is 2^{n-1} , where $n \geq 1$.
- The maximum number of nodes in a binary tree of height n is $2^n - 1$, where $n \geq 1$.
- For any non-empty tree, $n_l = n_d + 1$ where n_l is the number of leaf nodes and n_d is the number of nodes of degree 2.

41. What is meant by binary tree traversal?

Traversing a binary tree means moving through all the nodes in the binary tree,

visiting each node in the tree only once.

42. What are the different binary tree traversal techniques?

- Preorder traversal
- Inorder traversal
- Postorder traversal
- **Levelorder traversal**

43. What are the tasks performed during inorder traversal?

- Traverse the left sub-tree
- Process the root node
- Traverse the right sub-tree

44. What are the tasks performed during postorder traversal?

- Traverse the left sub-tree
- Traverse the right sub-tree
- Process the root node

45. State the merits of linear representation of binary trees.

- Storage method is easy and can be easily implemented in arrays
- When the location of a parent/child node is known, other one can be determined easily
- It requires static memory allocation so it is easily implemented in all programming language

46. State the demerit of linear representation of binary trees.

Insertions and deletions in a node take an excessive amount of processing time due to data movement up and down the array.

47. State the merit of linked representation of binary trees.

Insertions and deletions in a node involve no data movement except the rearrangement of pointers, hence less processing time.

48. State the demerits of linked representation of binary trees.

- Given a node structure, it is difficult to determine its parent node
- Memory spaces are wasted for storing null pointers for the nodes, which have one or no sub-trees
- It requires dynamic memory allocation, which is not possible in some programming language

49. Define a binary search tree

A binary search tree is a special binary tree, which is either empty or it should satisfy the following characteristics:

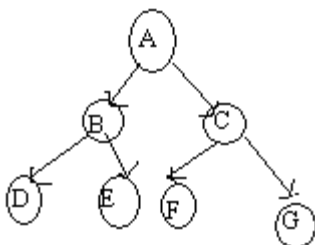
Every node has a value and no two nodes should have the same value i.e) the values in the binary search tree are distinct

- The values in any left sub-tree is less than the value of its parent node
- The values in any right sub-tree is greater than the value of its parent node
- The left and right sub-trees of each node are again binary search trees

50. What is the use of threaded binary tree?

In threaded binary tree, the NULL pointers are replaced by some addresses. The left pointer of the node points to its predecessor and the right pointer of the node points to its successor.

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

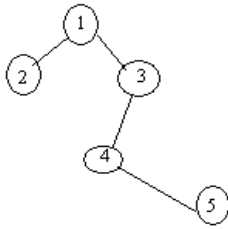


Inorder : D H B E A F C I G J

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

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

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



At location 6

1	2	3	-	-	4	-	-	5
---	---	---	---	---	---	---	---	---

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

52. Define AVL Tree.

An empty tree is height balanced. If T is a non-empty binary tree with TL and TR as its left and right subtrees, then T is height balanced if

- i) TL and TR are height balanced and
- ii) $hL - hR \leq 1$

Where hL and hR are the heights of TL and TR respectively.

53. What do you mean by balanced trees?

Balanced trees have the structure of binary trees and obey binary search tree properties. Apart from these properties, they have some special constraints, which differ from one data structure to another. However, these constraints are aimed only at reducing the height of the tree, because this factor determines the time complexity.

Eg: AVL trees, Splay trees.

54. What are the categories of AVL rotations?

Let A be the nearest ancestor of the newly inserted node which has the balancing factor ± 2 . Then the rotations can be classified into the following four categories:

Left-Left: The newly inserted node is in the left subtree of the left child of A.

Right-Right: The newly inserted node is in the right subtree of the right child of

A. Left-Right: The newly inserted node is in the right subtree of the left child of A.

Right-Left: The newly inserted node is in the left subtree of the right child of A.

55. What do you mean by balance factor of a node in AVL tree?

The height of left subtree minus height of right subtree is called balance factor of a node in AVL tree. The balance factor may be either 0 or +1 or -1. The height of an empty tree is -1.

56. Define Heap.

A heap is defined to be a complete binary tree with the property that the value of each node is at least as small as the value of its child nodes, if they exist. The root node of the heap has the smallest value in the tree.

57. What is the minimum number of nodes in an AVL tree of height h?

The minimum number of nodes $S(h)$, in an AVL tree of height h is given

by $S(h) = S(h-1) + S(h-2) + 1$. For $h=0$, $S(h)=1$.

58. Define B-tree of order M.

A B-tree of order M is a tree that is not binary with the following structural properties:

- The root is either a leaf or has between 2 and M children.
- All non-leaf nodes (except the root) have between $M/2$ and M children.
- All leaves are at the same depth.

59. Define Hashing.

Hashing is the transformation of string of characters into a usually shorter fixed length value or key that represents the original string. Hashing is used to index and retrieve items in a database because it is faster to find the item using the short hashed key than to find it using the original value.

60. What do you mean by hash table?

The hash table data structure is merely an array of some fixed size, containing the keys. A key is a string with an associated value. Each key is mapped into some number in the range 0 to $\text{tablesize}-1$ and placed in the appropriate cell.

61. What do you mean by hash function?

A hash function is a key to address transformation which acts upon a given key to compute the relative position of the key in an array. The choice of hash function should be simple and it must distribute the data evenly. A simple hash function is $\text{hash_key} = \text{key} \bmod \text{tablesize}$.

62. Write the importance of hashing.

- Maps key with the corresponding value using hash function.
- Hash tables support the efficient addition of new entries and the time spent on searching for the required data is independent of the number of items stored.

63. What do you mean by collision in hashing?

When an element is inserted, it hashes to the same value as an already inserted element, and then it produces collision.

64. What do you mean by separate chaining?

Separate chaining is a collision resolution technique to keep the list of all elements that hash to the same value. This is called separate chaining because each hash table element is a separate chain (linked list). Each linked list contains all the elements whose keys hash to the same index.

65. Write the advantage of separate chaining.

- More number of elements can be inserted as it uses linked lists.

66. Write the disadvantages of separate chaining.

- The elements are evenly distributed. Some elements may have more elements and some may not have anything.
- It requires pointers. This leads to slow the algorithm down a bit because of the time required to allocate new cells, and also essentially requires the implementation of a second data structure.

67. What do you mean by open addressing?

Open addressing is a collision resolving strategy in which, if collision occurs alternative cells are tried until an empty cell is found. The cells $h_0(x)$, $h_1(x)$, $h_2(x)$, are tried in succession, where $h_i(x) = (\text{Hash}(x) + F(i)) \bmod \text{Tablesize}$ with $F(0) = 0$. The function F is the collision resolution strategy.

68. What are the types of collision resolution strategies in open addressing?

- Linear probing
- Quadratic probing
- Double hashing

69. What do you mean by Probing?

Probing is the process of getting next available hash table array cell.

70. What do you mean by linear probing?

Linear probing is an open addressing collision resolution strategy in which F is a linear function of i , $F(i)=i$. This amounts to trying sequentially in search of an empty cell. If the table is big enough, a free cell can always be found, but the time to do so can get quite large.

71. List the limitations of linear probing.

- Time taken for finding the next available cell is large.
- In linear probing, we come across a problem known as clustering.

1. Define Graph.

A graph G consists of a nonempty set V which is a set of nodes of the graph, a set E which is the set of edges of the graph, and a mapping from the set of edges E to a set of pairs of elements of V . It can also be represented as $G=(V, E)$.

2. Define adjacent nodes.

Any two nodes which are connected by an edge in a graph are called adjacent nodes. For example, if an edge $x \in E$ is associated with a pair of nodes (u,v) where $u, v \in V$, then we say that the edge x connects the nodes u and v .

3. What is a directed graph?

A graph in which every edge is directed is called a directed graph.

4. What is an undirected graph?

A graph in which every edge is undirected is called an undirected graph.

5. What is a loop?

An edge of a graph which connects to itself is called a loop or sling.

6. What is a simple graph?

A simple graph is a graph, which has not more than one edge between a pair of nodes. Such a graph is called a simple graph.

7. What is a weighted graph?

A graph in which weights are assigned to every edge is called a weighted graph.

8. Define outdegree of a graph?

In a directed graph, for any node v , the number of edges which have v as their initial node is called the out degree of the node v .

9. Define indegree of a graph?

In a directed graph, for any node v , the number of edges which have v as their terminal node is called the indegree of the node v .

10. Define path in a graph?

The path in a graph is the route taken to reach terminal node from a starting node.

11. What is a simple path?

A path in a diagram in which the edges are distinct is called a simple path. It is also called as edge simple.

12. What is a cycle or a circuit?

A path which originates and ends in the same node is called a cycle or circuit.

13. What is an acyclic graph?

A simple diagram which does not have any cycles is called an acyclic graph.

14. What is meant by strongly connected in a graph?

An undirected graph is connected, if there is a path from every vertex to every other vertex. A directed graph with this property is called strongly connected.

15. When is a graph said to be weakly connected?

When a directed graph is not strongly connected but the underlying graph is connected, then the graph is said to be weakly connected.

16. Name the different ways of representing a graph?

- a. Adjacency matrix
- b. Adjacency list

17. What is an undirected acyclic graph?

When every edge in an acyclic graph is undirected, it is called an undirected acyclic graph. It is also called as undirected forest.

18. What are the two traversal strategies used in traversing a graph?

- a. Breadth first search
- b. Depth first search

19. What is a minimum spanning tree?

A minimum spanning tree of an undirected graph G is a tree formed from graph edges that connects all the vertices of G at the lowest total cost.

20. Name two algorithms to find minimum spanning tree

- Kruskal's algorithm
- Prim's algorithm

21. Define graph traversals.

Traversing a graph is an efficient way to visit each vertex and edge exactly once.

22. List the two important key points of depth first search.

- i) If path exists from one node to another node, walk across the edge – exploring

the edge.

ii) If path does not exist from one specific node to any other node, return to the previous node where we have been before – backtracking.

23. What do you mean by breadth first search (BFS)?

BFS performs simultaneous explorations starting from a common point and spreading out independently.

24. Differentiate BFS and DFS. No.	DFS	BFS
1.	Backtracking is possible from a dead end.	Backtracking is not possible
2.	Vertices from which exploration is incomplete are processed in a	The vertices to be explored are organized as a
3.	Search is done in one particular direction.	The vertices in the same level are maintained

25. What do you mean by tree edge?

If w is undiscovered at the time vw is explored, then vw is called a tree edge and v becomes the parent of w .

26. What do you mean by back edge?

If w is the ancestor of v , then vw is called a back edge.

27. Define biconnectivity.

A connected graph G is said to be biconnected, if it remains connected after removal of any one vertex and the edges that are incident upon that vertex. A connected graph is biconnected, if it has no articulation points.

28. What do you mean by articulation point?

If a graph is not biconnected, the vertices whose removal would disconnect the graph are known as articulation points.

29. What do you mean by shortest path?

A path having minimum weight between two vertices is known as shortest path, in which weight is always a positive number.

30. Define Activity node graph.

Activity node graphs represent a set of activities and scheduling constraints. Each node represents an activity (task), and an edge represents the next activity.

31. Define adjacency list.

Adjacency list is an array indexed by vertex number containing linked lists. Each node V_i the i^{th} array entry contains a list with information on all edges of G that leave V_i . It is used to represent the graph related problems.

UNIT V

1. Define an algorithm.

The algorithm is defined as a collection of unambiguous instructions occurring in some specific sequence and such an algorithm should produce output for given set of input in finite amount of time.

2. What is greedy algorithm?

In greedy method, the solution is constructed through a sequence of steps, each expanding a partially constructed solution obtained so far, until a complete solution to the problem is reached. At each step the choice made should be,

- **Feasible** – It should satisfy the problem's constraints.
- **Locally Optimal** – Among all feasible solutions the best choice is to be made.
- **Irrevocable** – Once the particular choice is made then it should not get changed on subsequent steps.

3. What are the applications of greedy method?

- a) Knapsack Problem.
- b) Prim's algorithm
- c) Kruskal's algorithm
- d) Finding shortest path

- e) Job sequencing with deadlines
- f) Optimal storage on tapes

4. Write general method of divide and conquer.

- In divide and conquer method, a given problem is,
 - i. Divided into smaller sub problems.
 - ii. These sub problems are solved independently.
 - iii. Combining all the solutions of sub problems into a solution of the whole.
- If the sub problems are large enough then divide and conquer is reapplied.
- The generated sub problems are usually of same type as the original problem.
- A control abstraction for divide and conquer is as given below using control abstraction a flow of control of a procedure is given.

5. Define Dynamic Programming

Dynamic Programming is typically applied to optimization problem. For each given problem, we may get any number of solutions from which we obtain for optimum solution (i.e. minimum value or maximum value solution.) And such an optimal solution becomes the solution to the given problem.

6. Comparison between divide and conquer and dynamic programming.

Sl. No.	Divide and conquer	Dynamic programming
1.	The problem is divided into small sub problems. These sub problems are solved independently. Finally all the solutions of sub problems are collected together to get the solution to the given problem.	In dynamic programming many decision sequences are generated and all the overlapping sub instances are considered.
2.	In this method duplications in sub solutions are neglected, i.e., duplicate sub solutions may be obtained.	In dynamic computing duplications in solutions is avoided totally.
3.	Divide and conquer is less efficient because of rework on solutions.	Dynamic programming is efficient than divide and conquer strategy.
4.	The divide and conquer uses top	Dynamic programming uses bottom

	down approach of problem solving.	up approach of problem solving.
5.	Divide and conquer splits its input at specific deterministic points usually in the middle.	Dynamic programming splits its input at every possible split points rather than at a particular point. After trying all split points it determines which split point is optimal.

7. Comparison between greedy algorithm and dynamic programming.

Sl. No.	Greedy Method	Dynamic Programming
1.	Greedy method is used for obtaining optimum solution.	Dynamic programming is also for obtaining optimum solution.
2.	In greedy method a set of feasible solutions and the picks up the optimum solution.	There is no special set of feasible solutions in this method.
3.	In greedy method optimum selection is without revising previously generated solutions.	Dynamic programming considers all possible sequences in order to obtain the optimum solution.
4.	In greedy method there is no as such guarantee of getting optimum solution.	It is guaranteed that the dynamic programming will generate optimal solution using principle of optimality.

8. What are the steps of dynamic programming?

- Characterize the structure of optimal solution. That means develop a mathematical notation that can express any solution and sub solution for the given problem.
- Recursively define the value of an optimal solution.
- By using bottom up technique compute value of optimal solution. For that you have to develop a recurrence relation that relates a solution to its sub solutions, using the mathematical notation of step 1.
- Compute an optimal solution from computed information.

9. State the principle of backtracking.

Backtracking is a method in which the desired solution is expressed as n tuple $(x_1, x_2, x_3, \dots, x_n)$ which is chosen from solution space, using backtrack formulation. The solution obtained i.e. $(x_1, x_2, x_3, \dots, x_n)$ can either minimizes or maximizes or satisfies the criteria function.

10. What is state space tree?

A state space tree is a rooted tree whose nodes represent partially constructed solutions to given problem. In backtracking method the state space tree is built for finding the solution. This tree is built using depth first search fashion.

11. How to write an algorithm?

Algorithm is basically a sequence of instructions written in simple English language. The algorithm is broadly divided into two sections.

Algorithm heading

It consists of name of algorithm, problem description, input and output.

Algorithm body

It consists of logical body of the algorithm by making use of various programming constructs and assignment statement.

12. What are the steps of algorithm design?

1. Understand the problem.
2. Decision making
 - a. Capabilities of computational devices
 - b. Select exact/approximate method
 - c. Data structures
 - d. Algorithmic strategies
3. Design of algorithm
4. Algorithm verification
5. Analysis of an algorithm
6. Coding of algorithm

13. What are the types of asymptotic notation?

- a. Big oh notation.
- b. Omega notation
- c. Theta notation

14. Define Big oh notation?

Let $F(n)$ and $g(n)$ be two non negative functions.

Let n_0 and constant c are two integers such that n_0 denotes some value of input and $n > n_0$ similarly c is some constant such that $c > 0$. We can write

$F(n) \leq c * g(n)$. then $F(n)$ is big oh of $g(n)$. It is also denoted as $F(n) \in O(g(n))$. In other words $F(n)$ is less than $g(n)$ is multiple of some constant c .

15. Define Recurrence equation.

The recurrence equation is an equation that defines a sequence recursively. It is normally in following form

$$T(n) = T(n-1) + n \dots\dots 1 \quad \text{for } n > 0$$

$$T(0) = 0 \quad \dots\dots 2$$

Here equation 1 is called recurrence relation and equation 2 is called initial condition. The recurrence equation can have infinite number of sequence. The general solution to the recursive function specifies some formula.

16. Define solving recurrence equation.

The recurrence relation can be solved by following methods

- a. Substitution method
- b. Master's method

17. What is the general plan for mathematical analysis of non recursive algorithm?

1. Decide the input size based on parameter n .
2. Identify algorithm's basic operation(s).

3. Check how many times the basic operation is executed. Then find whether the

Execution of basic operation depends upon the input size n . Determine worst, average, and best cases for input of size n . If the basic operation depends upon worst case, average case, and best case then that has to be analyzed separately.

4. Set up a sum for the number of times the basic operation is executed.
5. Simplify the sum using standard formula and rules.

18. What is the general plan for mathematical analysis of non recursive algorithm?

1. Decide the input size based on parameter n .
2. Identify algorithm's basic operation(s).
3. Check how many times the basic operation is executed. Then find whether

the

Execution of basic operation depends upon the input size n . Determine worst, average, and best cases for input of size n . If the basic operation depends upon worst case, average case, and best case then that has to be analyzed separately.

4. Set up the recurrence relation with some initial condition and expressing the basic operation.
5. Solve the recurrence relation.

19. Distinguish between backtracking and branch and bound techniques.

Sl. No.	Backtracking	Branch and Bound
1.	Solution for backtracking is traced using depth first search.	In this method, it is not necessary to use depth first search for obtaining the solution, even the breadth first search, best first search can be applied.
2.	Typically decision problems can be solved using backtracking.	Typically optimization problems can be solved using branch and bound.

3.	While finding the solution to the problem bad choices can be made.	It proceeds on better solutions. So there cannot be a bad solution.
4.	The state space tree is searched until the solution is obtained.	The state space tree needs to be searched completely as there may be chances of being an optimum solution any where in state space tree.
5.	Applications of backtracking are M coloring, knapsack.	Applications of branch and bound are job sequencing, TSP.

20. Define P and NP.

Problems that can be solved in polynomial time” (“P” stands for polynomial).

Eg. Searching of key element, sorting of elements, All pair shortest path.

NP stands for “non deterministic polynomial time”. Note that NP does not stand for “non polynomial”

Eg. Traveling salesman Problem, graph coloring problem, Knapsack problem, Hamiltonian circuit problem.

21. Two stages of non deterministic algorithm.

- a. **Non deterministic (Guessing) stage** → Generate an arbitrary string that can be thought of as a candidate solution.
- b. **Deterministic (“Verification”) stage** → In this stage it takes as input the candidate solution and the instance to the problem and returns yes if the candidate solution represents actual solution.