UNIT I C PROGRAMMING BASICS

Part A

1. Distinguish between high level language and low level language.

<table>
<thead>
<tr>
<th>LOW LEVEL</th>
<th>HIGH LEVEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low level language is machine readable form of program</td>
<td>high level language will be in human readable form</td>
</tr>
<tr>
<td>Low level language are difficult to write and compile</td>
<td>high level languages are easy to write as well as compile</td>
</tr>
<tr>
<td>Low level language are compact and require less memory space</td>
<td>High level language uses compilers and interpreters which requires large memory space</td>
</tr>
</tbody>
</table>

2. Tell the use of return type of printf() & scanf()
   - `printf()` returns the number of characters successfully written on the output
   - `scanf()` returns number of items successfully read.

3. What is a variable?
   - Variables are simply names used to refer to some location in memory – a location that holds a value with which we are working. It may help to think of variables as a placeholder for a value.

4. Distinguish the terms Break and Continue
   - The major difference between break and continue statements in C language is that a `break` causes the innermost enclosing loop or switch to be exited immediately. Whereas, the `continue` statement causes the next iteration of the enclosing for, while, or do loop to begin.

5. Explain the various form of looping statement.

<table>
<thead>
<tr>
<th>S.No</th>
<th>Desc</th>
</tr>
</thead>
</table>
| 1    | **While Loop**  
Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.  
| 2    | **for loop**  
Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.  
| 3    | **do...while loop**  
It is more like a while statement, except that it tests the condition at the end of the loop body.  
| 4    | **nested loops**  
You can use one or more loops inside any other while, for, or do while loop.  

6. Write the syntax of array declaration with an example
   An array is a collection of data that holds fixed number of values of same type.
   **For example:** if you want to store marks of 100 students, you can create an array for it.
   **Syntax:**
   ```
   data_type array_name[array_size];
   ```
   **Ex:**
   ```
   float marks[100];
   ```

7. Discuss the types of I/O statements available in C.
   **Unformatted I/O functions**
   a) getchar()
   b) putchar()
   c) gets()
   d) puts()
   e) getch()
   f) getche()
   **Formatted I/O functions**
   a) scanf()
   b) printf()

8. Show the declaration of a string.
   Strings are defined as an array of characters. The difference between a character array and a string is the string is terminated with a special character `\0`.
   **Syntax:**
   ```
   char str_name[size];
   ```

9. Identify the features of array.
   1) An array holds elements that have the same data type.
   2) Array elements are stored in subsequent memory locations.
   3) Two-dimensional array elements are stored row by row in subsequent memory locations.
   4) Array name represents the address of the starting element.
   5) Array size should be mentioned in the declaration. Array size must be a constant expression and not a variable.

10. Show the c code that narrates the difference between do-while and while loop
    **Do while:**
    ```
    int a=10;
    do{
    System.out.println("I am executing do while"+a);
    ```
}while(a<10);

While:
    int b=10;
    while(b<10)
    {
        System.out.println("I am executing while"+b);
    }

11. Define global declaration?
The variables that are used in more than one function throughout the program are called global variables and declared outside of all the function that is before the main () function.

12. Define data types?
Data type is the type of data that are going to access within the program. "C" supports different data types:

<table>
<thead>
<tr>
<th>Primary</th>
<th>Userdefined</th>
<th>Derived</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>typedef</td>
<td>pointers</td>
</tr>
<tr>
<td>int</td>
<td>float</td>
<td>structures</td>
</tr>
<tr>
<td>double</td>
<td></td>
<td>union</td>
</tr>
</tbody>
</table>

Example: int a,b; here int is data type

13. What is two dimensional array?
Two dimensional is an array of one dimensional array. The elements in the array are referenced with help of its row and column index.
Example: Int a[2][2];

14. What is decision making statement?
Decision making statement is used to break the normal flow of the program and execute part of the statement based on some condition.

15. Define Operator with example?
An operator is a symbol that specifies an operation to be performed on operands. Some operators require two operands called binary operators, while other acts upon only one operand called unary operator.
Example: a+b here a,b are operands and + is operator

16. Define conditional operator or ternary operator?
Conditional operator itself checks the condition and executes the statement depending on the condition. (a>b)? a:b if a is greater than b means the a value will be return otherwise b value will be return.
Example: big=a>b?a:b;
17. **Compare of switch() case and nested if statement**

<table>
<thead>
<tr>
<th>Switch</th>
<th>Nested if case</th>
</tr>
</thead>
<tbody>
<tr>
<td>The switch() can test only constant values.</td>
<td>The if can evaluate relational or logical expressions.</td>
</tr>
<tr>
<td>No two case statements have identical constants in the same switch</td>
<td>Same conditions may be repeated for number of times</td>
</tr>
<tr>
<td>Character constants are automatically converted to integers</td>
<td>Character constants are automatically converted to integers</td>
</tr>
</tbody>
</table>

18. **Define break statement?**

The break statements is used to terminate the loop. When the break statement is encountered inside a loop, the loop is immediately exited and the program continues with the statement immediately following the loop.

Example:
```
while(condition)
{
    ..........  
    if(condition)
    break;
    ....
}
```

19. **What is an Operator and Operand?**

An operator is a symbol that specifies an operation to be performed on operands.

Example: *, +, -, / are called arithmetic operators. The data items that operators act upon are called operands.

Example: a+b; In this statement a and b are called operands.

20. **What is the difference between scanf() and gets() function?**

In scanf() when there is a blank was typed, the scanf() assumes that it is an end. gets() assumes the enter key as end. That is gets() gets a new line (\n) terminated string of characters from the keyboard and replaces the \n with \0.

---

**Part-B**

1. Describe the structure of a C program with an example.
2. What is a two dimensional array explain its initialization?
3. Analyze the various string functions with example.
4. Formulate a C program to search an element from an array.
5. Develop a C program for the following:
   i) To find the area and circumference of the circle.
   ii) To find the sum of 100 integers.
6. i) Write a C program to find whether the given year is leap year or not.
   ii) Write a C program to find whether the given number is palindrome.
UNIT 2 FUNCTIONS, POINTERS, STRUCTURES AND UNIONS
PART-A

1. Define function?
Function is group of statements that can be performing a task. Function reduces the amount of coding and the function can be called from another program.

Example:
```c
main()
{
    fun();
    -------
}
```

2. Distinguish between Call by value Call by reference.

<table>
<thead>
<tr>
<th>Call by value</th>
<th>Call by reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>In call by value, the value of actual agreements is passed to the formal arguments and the operation is done on formal arguments.</td>
<td>In call by reference, the address of actual argument values is passed to formal argument</td>
</tr>
<tr>
<td>Formal arguments values are photocopies of actual arguments values.</td>
<td>Formal arguments values are pointers to the actual argument values.</td>
</tr>
<tr>
<td>Changes made in formal arguments valued do not affect the actual arguments values.</td>
<td>Since Address is passed, the changes made in the both arguments values are permanent</td>
</tr>
</tbody>
</table>

3. What is a Pointer? How a variable is declared to the pointer?
Pointer is a variable which holds the address of another variable.

**Pointer Declaration:**
```c
datatype *variablename;
```

**Example:**
```c
int *x; c=5; x=&a;
```

4. What are the uses of Pointers?
- Pointers are used to return more than one value to the function
- Pointers are more efficient in handling the data in arrays
- Pointers reduce the length and complexity of the program
- They increase the execution speed
- The pointers save data storage space in memory

5. What is the difference between if and while statement?

<table>
<thead>
<tr>
<th>If</th>
<th>While</th>
</tr>
</thead>
<tbody>
<tr>
<td>It is a conditional statement</td>
<td>It is a loop control statement</td>
</tr>
<tr>
<td>If the condition is true, it executes</td>
<td>Executes the statements within the</td>
</tr>
</tbody>
</table>
some statements. while block if the condition is true
If the condition is false then it Stops the execution the statements
If the condition is false the control is transferred to the next statement of the loop.

### 6. Define pre-processor in C.
Pre-processor are used to link the library files in to source program, that are placed before the main() function and it have three pre-processor directives that are
- Macro inclusion
- Conditional inclusion
- File inclusion

### 7. Define recursive function?
A function is a set of instructions used to perform a specified task which repeatedly occurs in the main program. If a function calls itself again and again that function is called recursive function.

### 8. Define null pointer?
A pointer is said to be a null pointer when its right value is 0, a null pointer can never point to a valid data. For checking a pointer, if it is assigned to 0, then it is a null pointer and is not valid

Example:
```c
int *a;
int *b;
a=b=0;
```

### 9. Compare array and structure

<table>
<thead>
<tr>
<th>Arrays</th>
<th>Structures</th>
</tr>
</thead>
<tbody>
<tr>
<td>An array is a collection of data items of same data type.Arrays can only be declared.</td>
<td>A structure is a collection of data items of different data types</td>
</tr>
<tr>
<td>There is no keyword for arrays.</td>
<td>The keyword for structures is struct</td>
</tr>
<tr>
<td>An array name represents the address of the starting element.</td>
<td>A structure name is known as tag. It is a Shorthand notation of the declaration.</td>
</tr>
<tr>
<td>An array cannot have bit fields.</td>
<td>A structure may contain bit fields</td>
</tr>
</tbody>
</table>

### 10. Compare structures and unions.

<table>
<thead>
<tr>
<th>Structures</th>
<th>Unions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Every member has its own memory</td>
<td>All members use the same memory.</td>
</tr>
<tr>
<td>The keyword used is struct.</td>
<td>The keyword used is Union.</td>
</tr>
<tr>
<td>Consumes more space compared to union</td>
<td>Consumes less space compared to structure</td>
</tr>
</tbody>
</table>
11. Define Structure in C.

A structure contains one or more data items of different data type in which the individual elements can differ in type. A simple structure may contain the integer elements, float elements and character elements etc. and the individual elements are called members.

Example:
```c
struct result
{
    int marks;
    float avg;
    char grade;
}std;
```

12. Rules for declaring a structure?

- A structure must end with a semicolon
- Usually a structure appears at the top of the source program.
- Each structure element must be terminated.
- The structure must be accessed with structure variable with dot (.) operator.

13. Define structure pointers

Pointer is a variable, it contain address of another variable and the structure pointers are declared by placing * in front of a structure variable’s name.

Example:
```c
struct result
{
    int marks;
    float avg;
    char grade;
};
struct result *sam;
```

14. Define union?

A union, is a collection of variables of different types, just like structure. Union is a derived data type and the difference between union and structure is in terms of storage. In structure each member has its own storage location, whereas all the members of union use the same memory location

Example:
```c
union result
{
    int marks;
    float avg;
    char grade;
}std;
```
15. What is the use of sizeof( ) operator?
   - The sizeof ( ) operator gives the bytes occupied by a variable.
   - No of bytes occupied varies from variable to variable depending upon its data types.
   Example:
     ```c
     int x,y;
     printf("%d",sizeof(x));
     ```
     Output:
     ```
     2
     ```

16. What is pointer arithmetic?
   C - Pointer arithmetic. A pointer in c is an address, which is a numeric value. Therefore, you can perform arithmetic operations on a pointer just as you can on a numeric value. There are four arithmetic operators that can be used on pointers: ++, --, +, and –

17. Illustrate the need of typedef.
   typedef is a keyword used in C language to assign alternative names to existing datatypes. It’s mostly used with user defined datatypes, when names of the datatypes become slightly complicated to use in programs. Following is the general syntax for using typedef,
   ```c
   typedef <existing_name> <alias_name>
   ```

18. Extend your views about malloc and calloc
   malloc()
   This function allocates a size byte of memory. It returns a pointer (*) to the first byte, or if there is an error, it returns NULL (to ensure that the situation is out of memory). The format follows as:
   ```c
   (type *) malloc(sizeof(type));
   ```

   calloc()
   It’s used to allocate storage. The real difference between these two, is that calloc() initializes all bytes in the allocation block to zero, because it’s used to reserve space for dynamic arrays. It’s written like this.
   ```c
   (type *) calloc(num, size);
   ```

19. Summarize on initializing Unions.
   When initializing a union, the initializer list must have only one member, which initializes the first member of the union unless a designated initializer is used
   ```c
   union { int x; char c[4]; } 
   u = {1},
   u2 = { .c={\'1\'} }; 
   ```
20. List out the use of library function.
   
   `<stdio.h>` Standard Input/Output functions
   `<stdlib.h>` Standard Utility functions
   `<string.h>` String handling functions
   `<time.h>` Date time functions
   `<math.h>` Mathematical Functions

PART-B

1. i) Interpret function declaration and function definition. (6)
   ii) Summarize examples for the above.
2. List out the operations performed by pointers with example.
3. Examine about structures and its operations.
4. Distinguish Unions and structures along with programming examples.
5. Explain a C program to implement employee payroll of a company along with explanation.
6. Examine nested structures with a sample C program of your own example.
UNIT III LINEAR DATA STRUCTURES
PART-A

1. What is meant by an abstract data type?
   An ADT is an object with a generic description independent of implementation
details. This description includes a specification of an components from which the object
is made and also behavioural details of objects.

2. What is a linked list?
   Linked list is a kind of series of data structures, which are not necessarily
Adjacent in memory. Each structure contains the element and a pointer to a record
containing its successor.

3. What is singly linked list?
   A singly linked list is a linked list, there exists only one link field in each and every
node and all nodes are linked together in some sequential manner and this type of
linked list is called singly linked list.

4. What is a doubly linked list?
   In a simple linked list, there will be one pointer named as NEXT POINTER to
point the next element, whereas in a doubly linked list, there will be two pointers one to
point the next element and the other to point the previous element location.

5. Define double circularly linked list?
   In a doubly linked list, if the last node or pointer of the list, point to the first
element of the list, then it is a circularly linked list.

6. Define Polynomial ADT
   A polynomial object is a homogeneous ordered list of pairs < exponent, coefficient>, where each coefficient is unique. Operations include returning
the degree, extracting the coefficient for a given exponent, addition, multiplication,
evaluation for a given input.

7. What are the types of Linear linked list?
   - Singly linked lists
   - Circular singly linked lists
   - Doubly linked lists
   - Circular doubly linked lists

8. What are advantages of Linked lists?
   - Linked lists are dynamic data structures
   - The size is not fixed
   - Data can store non continuous memory blocks
   - Insertion and deletion of nodes are easier and efficient
   - Complex applications

SAKTHLP AP/IT
9. How to implement stack using singly linked list
   Stack is an Last In First Out (LIFO) data structure. Here, elements are inserted from one end called push operation and the same elements are deleted from the same end called pop operation. So, using singly linked list stack operations are performed in the front or other way. We can perform rear end also.

10. Define Dqueue?
    Dqueue is also a data structure where elements can be inserted from both ends and deleted from both ends. To implement a dqueue operations using singly linked list operations performed insert_front, delete_front, insert_rear, delete_rear and display functions.

11. How to search an element in list.
    Searching can be initiated from first node and it is compared with given element one after the other until the specified key is found or until the end of the list is encountered.

12. Compare stack and queue.
    Stack – Represents the collection of elements in Last In First Out order. Operations includes testing null stack, finding the top element in the stack, removal of top most element and adding elements on the top of the stack.

    Queue - Represents the collection of elements in First In First Out order. Operations include testing null queue, finding the next element, removal of elements and inserting the elements from the queue. Insertion of elements is at the end of the queue. Deletion of elements is from the beginning of the queue.

13. How to create a new node, give with an example?
    // A linked list node
    struct Node
    {
      int data;
      struct Node *next;
    };

14. Define Push and Pop operations
    • push, which adds an element to the collection, and
    • pop, which removes the most recently added element that was not yet removed.
15. Evaluate the value of the expression \( ab + c \times d \) using stack.

<table>
<thead>
<tr>
<th>Infix Expression</th>
<th>Prefix Expression</th>
<th>PostfixExpression</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A + B )</td>
<td>(+ A B)</td>
<td>( A B +)</td>
</tr>
<tr>
<td>( A + B \times C)</td>
<td>(+ A \times B C)</td>
<td>( A B C \times +)</td>
</tr>
</tbody>
</table>

16. 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

17. Define a priority queue.

A 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.

18. Why you need a data structure?

A data structure helps you to understand the relationship of one data element with the other and organize it within the memory. Sometimes the organization might be simple and can be very clearly visioned. E.g.) List of names of months in a year – Linear Data Structure, List of historical places in the world- Non-Linear Data Structure. A data structure helps you to analyze the data, store it and organize it in a logical and mathematical manner.

19. State the difference between primitive and non-primitive data types.

Primitive data types are the fundamental data types. E.g.) int, float, double, char
Non-primitive data types are user defined data types. Eg) Structure, Union and enumerated data types.
20. List the applications of stacks
   - Towers of Hanoi
   - Reversing a string
   - Balanced parenthesis
   - Recursion using stack
   - Evaluation of arithmetic expressions

PART-B
1. Explain array based implementation of list with example.
2. Discuss in detail about linked list ADT with example.
3. List and explain the Queue ADT operation for insertion and deletion routine in linked list.
4. Describe about the implementation stack using linked list.
5. i) List the process of postfix valuation with an example.
   ii) List and define the balancing symbols with example.
6. A deque is a data structure consisting of a list of items, on which the following operations are possible:
   - Push (X,D): Insert item X on the front end of deque D.
   - Pop (D): remove the front item from deque D and return it.
   - Inject(x,D): Insert item X on the rear end of deque D.
   - Eject (D): Remove the rear item from deque that take O(1) time per operation.
   Combine the above mentioned operations and write a C code to formulate deque operations.
UNIT IV NON-LINEAR DATA STRUCTURES

PART-A

1. Definition of a Tree
A tree is a set of one or more nodes T such that:
   i. there is a specially designated node called a root
   ii. The remaining nodes are partitioned into n disjointed set of nodes $T_1, T_2, ..., T_n$ each of which is a tree.

2. What is the difference between height and depth of a tree?
   - The depth of a node is the number of edges from the node to the tree's root node. A root node will have a depth of 0.
   - The height of a node is the number of edges on the longest path from the node to a leaf. A leaf node will have a height of 0.

3. Define Binary tree and list its properties?
   A binary tree is a finite set of nodes that is either empty or consist a root node and two disjoint binary trees called the left subtree and the right subtree.
   **Properties of Full Binary Tree**
   - A binary tree of height $h$ with no missing node.
   - All leaves are at level $h$ and all other nodes have two children.
   - All the nodes that are at a level less than $h$ have two children each.

4. What are the advantages of Binary Tree?
   - Searching in Binary tree become faster.
   - Binary tree provides six traversals.
   - Two of six traversals give sorted order of elements.
   - Maximum and minimum elements can be directly picked up.
   - It is used for graph traversal and to convert an expression to postfix and prefix forms.

5. What are the two methods of binary tree implementation?
   Two methods to implement a binary tree are,
   a. Linear representation.
   b. Linked representation
6. Identify the differences between binary search with linear search.
   Linear search requires that the elements are iterable (e.g. via a linked list), but does not require the elements to be sorted. On average, you need to scan half of the elements to find a specific one.

   Binary search requires that the elements are sorted and requires that random access to the elements is possible in constant time (i.e. you cannot use a linked list, but have to use e.g. an array). With these preconditions, you need up to log2n iterations to find an element, n being the total number of elements.

7. List the applications of binary tree
   1. Manipulate hierarchical data.
   2. Make information easy to search (see tree traversal).
   3. Manipulate sorted lists of data.
   4. As a workflow for compositing digital images for visual effects.
   5. Router algorithms

8. Assess the different type of tree traversal.
   - Inorder Traversal: To get nodes of BST in non-increasing order, a variation of Inorder traversal where Inorder traversal’s reversed can be used.
   - Preorder Traversal: Preorder traversal is used to create a copy of the tree. Preorder traversal is also used to get prefix expression on of an expression tree.
   - Postorder Traversal: Postorder traversal is also useful to get the postfix expression of an expression tree

9. Explain about union operation.
   - Join two subsets into a single subset.
   - In this post, we will discuss an application of Disjoint Set Data Structure. The application is to check whether a given graph contains a cycle or not.

10. Construct an acyclic graph.
11. Compare DFS and BFS.

<table>
<thead>
<tr>
<th></th>
<th>BFS</th>
<th>DFS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BFS starts traversal from the root node and then explore the</td>
<td>DFS starts the traversal from the root node and explore the</td>
</tr>
<tr>
<td></td>
<td>search in the level by level manner i.e. as close as possible</td>
<td>search as far as possible from the root node i.e. depth wise.</td>
</tr>
<tr>
<td></td>
<td>from the root node.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Breadth First Search can be done with the help of queue i.e.</td>
<td>Depth First Search can be done with the help of Stack i.e.</td>
</tr>
<tr>
<td></td>
<td>FIFO implementation.</td>
<td>LIFO implementations.</td>
</tr>
<tr>
<td></td>
<td>BFS requires more memory compare to DFS.</td>
<td>DFS require less memory compare to BFS.</td>
</tr>
<tr>
<td></td>
<td>This algorithm works in single stage. The visited vertices</td>
<td>This algorithm works in two stages – in the first stage the</td>
</tr>
<tr>
<td></td>
<td>are removed from the queue and then displayed at once.</td>
<td>visited vertices are pushed onto the stack and later on when</td>
</tr>
<tr>
<td></td>
<td></td>
<td>there is no vertex further to visit those are popped-off.</td>
</tr>
</tbody>
</table>

12. Explain the tree and graph.

Graph:
A graph consists of a set of nodes and a set of edges. An edge is a pair of nodes that are connected. A path is the term used to describe travelling between nodes that share an edge.

Tree:
A tree data structure, like a graph, is a collection of nodes. There is a root node. The node can then have children nodes. The children nodes can have their own children nodes called grandchildren nodes.

In graph theory, a connected component (or just component) of an undirected graph is a sub graph in which any two vertices are connected to each other by paths, and which is connected to no additional vertices in the super graph. For example, the graph shown in the illustration has three connected components.

Vertex with no incident edges is itself a connected component. A graph that is itself connected has exactly one connected component, consisting of the whole graph.
14. Label the different types of union.

Finite unions:
One can take the union of several sets simultaneously. For example, the union of three sets A, B, and C contains all elements of A, all elements of B, and all elements of C, and nothing else. Thus, x is an element of $A \cup B \cup C$ if and only if x is in at least one of A, B, and C.

Arbitrary unions:
The most general notion is the union of an arbitrary collection of sets, sometimes called an infinitely union. If M is a set or class whose elements are sets, then x is an element of the union of M if and only if there is at least one element A of M such that x is an element of A.

15. Define Graph and Acyclic graph.

Graph:
A graph consists of a set of nodes and a set of edges. An edge is a pair of nodes that are connected. A path is the term used to describe travelling between nodes that share an edge.

Acyclic Graph:
An acyclic graph is a graph having no graph cycles. Acyclic graphs are bipartite. A connected acyclic graph is known as a tree, and a possibly disconnected acyclic graph is known as a forest (i.e., a collection of trees).

16. List the different ways of representing graph.
1. Map over the vertices $v \in V$.
2. Map over the edges $(u,v) \in E$.
3. Return the degree of a vertex $v \in V$.
4. Determine if the edge $(u,v)$ is in $E$.
5. Insert or delete vertices.
6. Insert or delete edges.

17. List out few of the Application of tree data-structure?
- The manipulation of Arithmetic expression
- Used for Searching Operation
- Used to implement the file system of several popular operating systems
- Symbol Table construction
- Syntax analysis

18. Define expression tree?
Expression tree is also a binary tree in which the leafs terminal nodes or operands and non-terminal intermediate nodes are operators used for traversal.
19. Define lazy deletion?

When an element is to be deleted it is left in the tree itself and marked as being deleted. This is called as lazy deletion and is an efficient procedure if duplicate keys are present in the binary search tree, because the field that keeps count of the frequency of appearance of the element can be decremented of the element can be decremented.

20. Define complete binary tree.

If all its levels, possible except the last, have maximum number of nodes and if all the nodes in the last level appear as far left as possible.

**PART-B**

1. Write short note on the following terms related to tree:
   i) Path (2)
   ii) Degree (3)
   iii) Level (2)
   iv) Leaves (2)
   v) Child (2)
   vi) Height (2)

2. Conclude the types of tree traversal methods? Explain it with example and deduce a routine for each of them.

3. i) Illustrate your understanding by finding the inorder, preorder and postorder form for the following graph: (7)

   ![Graph Image]

   i) Show some applications of trees.

4. Analyze in detail the implementation of Binary Search Tree and perform its operations.

5. Describe in detail about the smart union algorithm.

6. i) Develop the following graph using depth first search. (7)

   ![Graph Image]

   ii) Construct a C routine for BFS and DFS.
UNIT V SEARCHING AND SORTING ALGORITHMS

1. What is meant by Sorting?
   Sorting is ordering of data in an increasing or decreasing fashion according to some linear relationship among the data items.

2. List the different sorting algorithms.
   • Bubble sort
   • Selection sort
   • Insertion sort
   • Shell sort
   • Quick sort
   • Radix sort
   • Heap sort
   • Merge sort

3. Why bubble sort is called so?
   The bubble sort gets its name because as array elements are sorted they gradually “bubble” to their proper positions, like bubbles rising in a glass of soda.

4. State the logic of bubble sort algorithm.
   The bubble sort repeatedly compares adjacent elements of an array. The first and second elements are compared and swapped if out of order. Then the second and third elements are compared and swapped if out of order. This sorting process continues until the last two elements of the array are compared and swapped if out of order.

5. What number is always sorted to the top of the list by each pass of the Bubble sort algorithm?
   Each pass through the list places the next largest value in its proper place. In essence, each item “bubbles” up to the location where it belongs.

6. When does the Bubble Sort Algorithm stop?
   The bubble sort stops when it examines the entire array and finds that no "swaps" are needed. The bubble sort keeps track of the occurring swaps by the use of a flag.

7. State the logic of selection sort algorithm.
   It finds the lowest value from the collection and moves it to the left. This is repeated until the complete collection is sorted.
8. What is the output of selection sort after the 2\textsuperscript{nd} iteration given the following sequence?

Numbers: 16 3 46 9 28 14
Answer: 3 9 46 16 28 14

9. How does insertion sort algorithm work?
   In every iteration an element is compared with all the elements before it. While comparing if it is found that the element can be inserted at a suitable position, then space is created for it by shifting the other elements one position up and inserts the desired element at the suitable position. This procedure is repeated for all the elements in the list until we get the sorted elements.

10. How many key comparisons and assignments an insertion sort makes in its worst case?
   The worst case performance in insertion sort occurs when the elements of the input array are in descending order. In that case, the first pass requires one comparison, the second pass requires two comparisons, third pass three comparisons,…..kth pass requires (k-1), and finally the last pass requires (n-1) comparisons. Therefore, total numbers of comparisons are:

   \[ f(n) = 1+2+3+\ldots\ldots+(n-k)+\ldots\ldots+(n-2)+(n-1) = \frac{n(n-1)}{2} = O(n^2) \]

11. Which sorting algorithm is easily adaptable to singly linked lists? Why?
   Insertion sort is easily adaptable to singly linked list. In this method there is an array link of pointers, one for each of the original array elements. Thus the array can be thought of as a linear link list pointed to by an external pointer first initialized to 0. To insert the k\textsuperscript{th} element the linked list is traversed until the proper position for x[k] is found, or until the end of the list is reached. At that point x[k] can be inserted into the list by merely adjusting the pointers without shifting any elements in the array which reduces insertion time.

12. Why Shell Sort is known diminishing increment sort?
   The distance between comparisons decreases as the sorting algorithm runs until the last phase in which adjacent elements are compared. In each step, the sorted number of the sequence is increased, until in the last step it is completely sorted.

13. What is the output of quick sort after the 3\textsuperscript{rd} iteration given the following sequence?

   24 56 47 35 10 90 82 31
   Pass 1:- (10) 24 (56 47 35 90 82 31)
   Pass 2:- 10 24 (56 47 35 90 82 31)
   Pass 3:- 10 24 (47 35 31) 56 (90 82)
14. **Mention the different ways to select a pivot element.**

The different ways to select a pivot element are

- Pick the first element as pivot
- Pick the last element as pivot
- Pick the Middle element as pivot
- Median-of-three elements
  - Pick three elements, and find the median x of these elements
  - Use that median as the pivot.
- Randomly pick an element as pivot.

15. **What is divide-and-conquer strategy?**

- Divide a problem into two or more sub problems
- Solve the sub problems recursively
- Obtain solution to original problem by combining these solutions

16. **Compare quick sort and merge sort.**

Quick sort has a best-case linear performance when the input is sorted, or nearly sorted. It has a worst-case quadratic performance when the input is sorted in reverse, or nearly sorted in reverse.

Merge sort performance is much more constrained and predictable than the performance of quick sort. The price for that reliability is that the average case of merge sort is slower than the average case of quick sort because the constant factor of merge sort is larger.

17. **Define Searching.**

Searching for data is one of the fundamental fields of computing. Often, the difference between a fast program and a slow one is the use of a good algorithm for the data set. Naturally, the use of a hash table or binary search tree will result in more efficient searching, but more often than not an array or linked list will be used. It is necessary to understand good ways of searching data structures not designed to support efficient search.

18. **What is linear search?**

In Linear Search the list is searched sequentially and the position is returned if the key element to be searched is available in the list, otherwise -1 is returned. The search in Linear Search starts at the beginning of an array and move to the end, testing for a match at each item.
19. What is Binary search?
A binary search, also called a dichotomizing search, is a digital scheme for locating a specific object in a large set. Each object in the set is given a key. The number of keys is always a power of 2. If there are 32 items in a list, for example, they might be numbered 0 through 31 (binary 00000 through 11111). If there are, say, only 29 items, they can be numbered 0 through 28 (binary 00000 through 11100), with the numbers 29 through 31 (binary 11101, 11110, and 11111) as dummy keys.

20. Define hash function?
Hash function takes an identifier and computes the address of that identifier in the hash table using some function. Hashing is a technique used for performing insertions, deletions, and finds in constant average time.

PART B

1. Write an algorithm to implement Bubble sort with suitable example.
2. Explain any two techniques to overcome hash collision.
3. Write an algorithm to implement insertion sort with suitable example.
4. Write an algorithm to implement selection sort with suitable example.
5. Write an algorithm to implement radix sort with suitable example.
6. Write an algorithm for binary search with suitable example.
7. Discuss the common collision resolution strategies used in closed hashing system.
8. Given the input { 4371, 1323, 6173, 4199, 4344, 9679, 1989 } and a hash function of h(X)=X (mod 10) show the resulting:
   a. Separate Chaining hash table
   b. Open addressing hash table using linear probing
9. Explain Re-hashing and Extendible hashing.
10. Show the result of inserting the keys 2,3,5,7,11,13,15,6,4 into an initially empty extendible hashing data structure with M=3. (8) (Nov 10)