Question 1) What do you mean by a Data structure?
Ans - Data Structure is a specialized means of organizing and storing data in computers in such a way that we can perform operations on the stored data more efficiently.

Question 2) What are some of the applications of DS?
Ans - The applications of DS are :

1. Array
2. Stack
3. Queue
4. LinkedList
5. Graph
6. Tree
7. Hash

Question 3) Write the syntax in C to create a node in the singly linked list.
Ans - Struct Node
{
Int demo;
Node *next;
};

Question 4) What are the different types of traversal techniques in a tree?
Ans - The types of Tree traversal are -

1. Preorder
2. Postorder
3. Inorder

Question 5) What are the applications of Graph DS?
Ans - Application of Graphs:

1. In Computer science graphs are used to represent the flow of computation.
2. In World Wide Web, web pages are considered to be the vertices. This is an example of Directed graph. It was the basic idea behind Google Page Ranking Algorithm.
3. In the Operating System, we come across the Resource Allocation Graph where each process and resources are considered to be vertices.

Question 6) Can we apply the Binary search algorithm to a sorted Linked list?
Ans - Yes, Binary search is possible on the linked list if the list is ordered and you know the count of elements in the list. But While sorting the list, you can access a single element at a time through a pointer to that node i.e. either a previous node or next node.

Question 7) When can you tell that a Memory Leak will occur?
Ans - Memory leak occurs when programmers create a memory in heap and forget to delete it. Memory leaks are particularly serious issues for programs like daemons and servers which by definition never terminate.

Question 8) How will you check if a given Binary Tree is a Binary Search Tree or not?
Ans- A Binary Search Tree (BST) is a binary tree with the following properties:

- The left subtree of a particular node will always contain nodes whose keys are less than that node's key.
- The right subtree of a particular node will always contain nodes with keys greater than that node's key.
- The left and right subtree of a particular node will also, in turn, be binary search trees.

Question 9) Which data structure is ideal to perform recursion operation and why?
Ans - Stack has the LIFO (Last In First Out) property; it remembers it's 'caller'. Therefore, it knows to whom it should return when the function has to return. On the other hand, recursion makes use of the system stack for storing the return addresses of the function calls.

Question 10) What are some of the most important applications of a Stack?
Ans - 1. Expression Handling
- Infix to Postfix or Infix to Prefix Conversion
- Postfix or Prefix Evaluation

2. Backtracking Procedure

Question 11) Convert the below given expression to its equivalent Prefix And Postfix notations?
Ans - Prefix expression is those expressions which have operators before the operands.
+AB.

Postfix expressions are those expressions which have operators after operands in the expressions.

Example: AB/

Prefix to postfix-
input /+XY+NM
Output: XY+NM+/

Question 12) Program to reverse a queue.
Ans-

```
import java.util.LinkedList;
import java.util.Queue;
import java.util.Stack;

public class Queue_reverse {
  static Queue<Integer> queue;
  static void Print()
  {
    while (!queue.isEmpty()) {
      System.out.print( queue.peek() + ", ");
      queue.remove();
    }  }

  // Function for reversing the queue
  static void reverse()
  {
    Stack<Integer> stack = new Stack<>();
    while (!queue.isEmpty()) {
      stack.add(queue.peek());
      queue.remove();
```

```
    }
    while (!stack.isEmpty()) {
      queue.add(stack.peek());
      stack.pop();
    }
  }
  public static void main(String args[])
  {
    queue = new LinkedList<Integer>();
    queue.add(10);
    queue.add(20);
    queue.add(30);
    queue.add(40);
    queue.add(50);
    queue.add(60);
    reverse();
    Print();
  }}
```

Question 13) Program to return the nth node from the end in a linked list.
Solution -

```
class LinkedList {
  Node head;

  class Node {
    int data;
    Node next;
    Node(int d)
    {
      data = d;
      next = null;
    }
  }
  void printNthFromLast(int n){
    int len = 0;
    Node temp = head;

    while (temp != null) {
      temp = temp.next;
      len++;
    }
    if (len < n)
      return;
     temp = head;

    for (int i = 1; i < len - n + 1; i++)
      temp = temp.next;
```

```
    System.out.println(temp.data);
  }
  public void push(int new_data)
  {
    Node new_node = new Node(new_data);
    new_node.next = head;
    head = new_node;
}
  public static void main(String[] args)
  {
    LinkedList list = new LinkedList();
    list.push(20);
    list.push(4);
    list.push(15);
    list.push(35);

    list.printNthFromLast(4);
  }}
```

Question -14) what are the advantages of a Linked list over an Array?

Ans - The advantages of a linked list over an array are:

1. Less memory consumption as only that many nodes are consumed as many values are to be stored.
2. Easy of Insertion/deletion only modification of pointers required.
3. Linked list dynamic in size whereas for arrays it is fixed size.
4. Size is not an issue as compared to arrays.

Question 15) What is the use of a doubly-linked list when compared to that of a singly-linked list?

Ans - Doubly linked list allows element two traversal. It can also be used to implement stacks as well as heaps and binary trees. If we need better performance while searching and memory is not a limitation in this case DLL is more preferred. The delete operation in DLL is more efficient if a pointer to the node to be deleted is given. In singly linked list, to delete a node, a pointer to the previous node is needed. To get this previous node, sometimes the list is traversed. In DLL, we can get the previous node using the previous pointer.

Question 16) What is the difference between an array and a stack?

Ans) Stack - Stack are based on Last In First Out principle. Insertion and deletion in stacks takes place only from one end of the list called the top.Stack has a dynamic size. Stack can contain elements of different data types.

Array - Array contains elements of same data type.Array has a fixed size.Insertion and deletion in array can be done at any index in the array.In the array the elements belong to indexes, i.e., if you want to get into the fourth element you have to write the variable name with its index or location within the square bracket.

Question 17) Reverse a linked list ?

Ans -

```
class LinkedList {
static Node head;
```

```
static class Node {
int data;
Node next;
Node(int d)
{
data = d;
next = null;
}
}
Node reverse(Node node)
{
Node prev = null;
Node current = node;
Node next = null;
while (current != null) {
next = current.next;
current.next = prev;
prev = current;
current = next;
}
node = prev;
return node;
}
void printList(Node node)
{
while (node != null) {
System.out.print(node.data + " ");
node = node.next;
}
}
public static void main(String[] args)
{
LinkedList list = new LinkedList();
list.head = new Node(85);
list.head.next = new Node(15);
list.head.next.next = new Node(4);
list.head.next.ne xt.next = new Node(20);
System.out.println("Given Linked list");
list.printList(head);
head = list.reverse(head);
System.out.println("");
System.out.println("Reversed linked list ");
list.printList(head);
}
}
```

Question - 18) Sorting a stack using a temporary stack

Ans -

```
import java.util.*;
class SortStack
{
   public static Stack<Integer> sortstack(Stack<Integer> input)
   {
      Stack<Integer> tmpStack = new Stack<Integer>();
      while(!input.isEmpty())
      {   int tmp = input.pop();
         while(!tmpStack.isEmpty() && tmpStack.peek() > tmp)
        {
         input.push(tmpStack.pop());
         }
        tmpStack.push(tmp);
      }
      return tmpStack;
   }
   public static void main(String args[])
   {
      Stack<Integer> input = new Stack<Integer>();
      input.add(12);
      input.add(45);
      input.add(68);
      input.add(96);
      input.add(29);
      input.add(78);
      Stack<Integer> tmpStack=sortstack(input);
      System.out.println("Sorted numbers are:");
      while (!tmpStack.empty())
      {
         System.out.print(tmpStack.pop()+" ");
      }
   }
}
```

Question - 19) Program to reverse first k elements of a queue?

Ans-

```
import java.util.LinkedList;
import java.util.Queue;
import java.util.Stack;

public class Reverse_k_element_queue {
   static Queue<Integer> queue;
   static void reverseQueueFirstKElements(int k) {
      if (queue.isEmpty() == true || k > queue.size())
         return;
      if (k <= 0)
         return;
```

```
        Stack<Integer> stack = new Stack<Integer>();
        for (int i = 0; i < k; i++) {
            stack.push(queue.peek());
            queue.remove();
        }
        while (!stack.empty()) {
            queue.add(stack.peek());
            stack.pop();  }
        for (int i = 0; i < queue.size() - k; i++) {
            queue.add(queue.peek());
            queue.remove();    }
    }static void Print() {
        while (!queue.isEmpty()) {
            System.out.print(queue.peek() + " ");
            queue.remove();
        } }
    public static void main(String args[]) {
        queue = new LinkedList<Integer>();
        queue.add(10);
        queue.add(20);
        queue.add(30);
        queue.add(40);
        queue.add(50);
        queue.add(60);
        queue.add(70);
        int k = 5;
        reverseQueueFirstKElements(k);
        Print();
    }
}
```

Question - 20)Replace each element of the array by its rank in the array?

Ans -

```
import java.util.*;

class Apoorv {
  static void changeArr(int[] input)
    {
        int newArray[]
            = Arrays.copyOfRange(input,0,input.length);
        Arrays.sort(newArray);
        int i;
        Map<Integer, Integer> ranks   = new HashMap<>();
        int rank = 1;

        for (int index = 0; index < newArray.length; index++) {
            int element = newArray[index];
            if (ranks.get(element) == null) {
                ranks.put(element, rank);
```

```
            rank++;
          }
        }
        for (int index = 0;
             index < input.length;
             index++) {
          int element = input[index];
          input[index] = ranks.get(input[index]);
        }
    }
    public static void main(String[] args)
    {
        int[] arr = { 87, 29, 197, 34 };
        changeArr(arr);
        System.out.println(Arrays.toString(arr));
    }
}
```

Question 21) Check if a given graph is a tree or not?
Ans -

```
import java.io.*;
import java.util.*;

class Graph
{
    private int V;
    private LinkedList<Integer> adj[];
      Graph(int v)
    {
        V = v;
        adj = new LinkedList[v];
        for (int i=0; i<v; ++i)
            adj[i] = new LinkedList();
    }
    void addEdge(int v,int w)
    {
        adj[v].add(w);
        adj[w].add(v);
    }
    Boolean isCyclicUtil(int v, Boolean visited[], int parent)
    {
        visited[v] = true;
        Integer i;
        Iterator<Integer> it = adj[v].iterator();
        while (it.hasNext())
        {
            i = it.next();
            if (!visited[i])
```

```
        {
            if (isCyclicUtil(i, visited, v))
                return true;
        }
         else if (i != parent)
          return true;    }
    return false;
  }
   Boolean isTree()
  {
    Boolean visited[] = new Boolean[V];
    for (int i = 0; i < V; i++)
      visited[i] = false;
    if (isCyclicUtil(0, visited, -1))
      return false;
    for (int u = 0; u < V; u++)
      if (!visited[u])
        return false;
    return true;
  }
  public static void main(String args[])
  {
    Graph g1 = new Graph(5);
    g1.addEdge(1, 0);
    g1.addEdge(0, 2);
    g1.addEdge(0, 3);
    g1.addEdge(3, 4);
    if (g1.isTree())
      System.out.println("Graph is Tree");
    else
      System.out.println("Graph is not Tree");

    Graph g2 = new Graph(5);
    g2.addEdge(1, 0);
    g2.addEdge(0, 2);
    g2.addEdge(2, 1);
    g2.addEdge(0, 3);
    g2.addEdge(3, 4);
    if (g2.isTree())
      System.out.println("Graph is Tree");
    else
      System.out.println("Graph is not Tree");

  } }
```

Question - 22) Find out the Kth smallest element in an unsorted array?

Ans -

```
import java.util.Arrays;
import java.util.Collections;
```

```
class Apoorv {
   public static int kthSmallest(Integer[] arr,int k)
   {
      Arrays.sort(arr);
  return arr[k - 1];
   }
   public static void main(String[] args)
   {
      Integer arr[] = new Integer[] { 32, 76, 98, 23, 128 };
      int k = 2;
      System.out.print("K’th smallest element is " + kthSmallest(arr, k));
   }
}
```

Question - 23) How to find the shortest path between two vertices?
Ans -

Question - 24) What is the minimum number of Queues needed to implement the priority queue?
Ans - Priority queues are applied using 2-D array where it has two rows one for element and second for priority ,so minimum numbers of queues are needed to implement two.

Question 25) Why is it said that searching a node in a binary search tree is efficient than that of a simple binary tree?
Ans - Binary Tree is a hierarchical data structure in which a child can have zero, one, or maximum two child nodes; each node contains a left pointer, a right pointer and a data element. There’s no particular order to how the nodes should be organized in the tree. Binary Search Tree, on the other hand, is an ordered binary tree in which there is a relative order to how the nodes should be organized.