**Q1. What do you mean by a Data structure?**

Ans- Data Structure is a way of collecting and organising data in such a way that we can perform operations on these data in an effective way. It is a way that specifies that how to organize and manipulate the data.

**Q2. What are some applications of DS?**

Ans- Data structures gives us a way to store and manage large amounts of data efficiently for uses such as large databases and cloud services. Because of this it is considered that, efficient data structures are key to designing efficient algorithms.

**Q3. What are the advantages of a Linked list over an array?**

Ans- The main and major benefit of a linked list over a conventional array is that the list elements can be easily inserted or removed without reallocation or reorganization of the entire structure. We can also insert and remove any node at any point in linked list with just three steps, but in arrays we have reallocate all the elements when an element is inserted or deleted. Linked list is dynamic and it can store generic objects but arrays cannot.

**Q4. Write the syntax in C to create a node in the singly linked list.**

Ans-
```
struct node{
        int data;
        struct node *next;
};
```

**Q5. What is the use of a doubly-linked list when compared to that of a single linked list?**

Ans- A doubly linked list can be traversed bidirectionally therefore it saves the overhead of traversal in cases when we roughly know the place of an element. Singly linked list is preferred when we need to save memory and searching is not required as pointer of single index is stored.

**Q6. What is the difference between an array and stack?**

Ans- The main difference between an array and stack is how they return the values, in an array we can access an index of an element and get the data whereas in a stack, the data is returned in a first in last out fashion, that is that the data that went in first is the last in order when we get data from the stack.

**Q7. What are the minimum number of Queues needed to implement the priority queue?**

Ans- We need two queues to implement the priority queue. One is used for storing the data, another is used for priorities. Priority queues are implemented 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 are two.

**Q8. What are the different types of traversal techniques in a tree?**

Ans- There are three types of traversal techniques in trees.
Inorder (Left, Root, Right)
Preorder (Root, Left, Right)
Postorder (Left, Right, Root)

**Q9. Why it is said that searching a node in a binary search tree is efficient than that of a simple binary tree?**

Ans- Searching a node in binary search tree is efficient because of property of binary search tree that each left side node is lesser than root node and each right side node is greater than root node so because of this searching will always follow a particular single path there is no need to traverse each node like binary tree.

**Q10. What are the applications of Graph DS?**

Ans- Graph DS is used to model scenarios where a component has relations with multiple components. Graph DS is used for modelling networks such as social media networks etc.

**Q11. Can we use binary search algorithm to a sorted linked list?**

Ans- We cannot apply binary search algorithm to a sorted linked list because the binary search heavily relies on indexes , and the overhead to bootstrap an index system defeats the purpose.

**Q12. When can you tell that a Memory Leak will occur?**

Ans- It may occur when the memory allocated to the objects which are not used or not of any use anymore is not released. It can also occur when an object which is already stored cannot be accessed by the code.

**Q13. How will you check if a given Binary Tree is a Binary Search Tree or not?**

Ans- There are three conditions a Binary Search Tree should satisfy. They are:
All nodes in the left sub-tree of a node (including parent) have values less than the node's value.
All nodes in the right sub-tree of a node (including parent) have values greater than the node's value.
Both left and right sub-trees are also binary search trees.

**Q14. Which data structure is ideal to perform recursion operation and why?**

Ans- Stack is used to perform recursion operation Because of its LIFO (Last in First Out) property it remembers its 'caller' so knows whom to return when the function has to return. Recursion makes use of system stack for storing the return addresses of the function calls. Every recursive function has its equivalent iterative (non-recursive) function.

**Q15. What are some of the most important applications of a Stack?**

Ans- 1. Infix to Postfix or Infix to Prefix Conversion. 2. Postfix or Prefix Evaluation. 3. Backtracking Procedure. 4. Recursion Procedure.

**Q16. Incomplete Question.**

**Q17. Sorting a stack using a temporary stack**

Ans-

```
public Stack<Integer> sortStack(Stack<Integer> stack){
   Stack<Integer> temp = new Stack<>();
    while(!stack.isEmpty()){
    int currentVal = stack.pop();
    if(temp.isEmpty() || temp.peek() < currentVal){
temp.push(currentVal);
      }
else{
          while(temp.peek()>currentVal){
stack.push(temp.pop());
          }
          temp.push(currentVal);
      }
 }
   while(!temp.isEmpty()){
       stack.push(temp.pop());
   }
   return stack;
}
```

**Q18: Program to reverse a queue**

Ans-

```
public Queue<Integer> reverseQueue(Queue<Integer> s){
Stack<Integer> stack = new Stack<>();
while(!s.isEmpty()){ stack.push(s.poll());
}
while(!stack.isEmpty()){
s.add(stack.pop());
```

```
} return s;
}
```

**Q19: Program to reverse first k elements of a queue**

Ans-

```
public Queue<Integer> reverseK(Queue<Integer> queue, int k){
Queue<Integer> newQueue = new LinkedList<>();
Stack<Integer> stack = new Stack<>();
while(k-- >0 ){
stack.push(queue.poll());
}
while(!stack.isEmpty()){
newQueue.add(stack.pop());
}
while(!queue.isEmpty()){
newQueue.add(queue.poll());
} return newQueue;
}
```

**Q20: Program to return the nth node from the end in a linked list.**

Ans-

```
public int getLength(Node n,int k){
if(n == null){
 return k;
}
return getLength(n.next, k+1);
}
public int nthNode(Node n, int nth){
 int getLengthFromStart = getLength(n,0) – nth;
int ret = 0;
Node current = n; while(getLengthFromStart-- > 0){
```

```
ret = Node.data; current = current.next;
} return ret;
}
```

**Q21 : Reverse a linked list**

Ans-

```
class Node{
   Node next;
   int i;
   public Node(){
     next = null;
   }
   public Node(int i){
       this.i = i;
        this.next = null;     }
 }
public Node reverseNode(Node n){
   Node prev = null;
   Node current = n;
   while(n != null){
       Node revNode = new Node();
        revNode.i = n.i;
       revNode.next = prev;
       prev = revNode;
       n = n.next;
   }
   return prev;
}
```

**Q22: Replace each element of the array by its rank in the array.**

Ans-

```
 static void changeArr(int[] input)
   {
      int newArray[]  = new int[input.length];
 for(int I = 0 ; I < input.length ; i++){
newArray[i] = input[i];
}
      Arrays.sort(newArray);
       int i = 0;
      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]);
      }
   }
```

**Q23: Check if a given graph is a tree or not?**

Ans- To check if a given graph is a tree or not, we have to check if the graph contains a cycle. A graph is a tree if it doesn't contain a cycle. Since it is not mentioned if it is a directed or un-directed graph, a particular cycle detection algorithm can't be prescribed but techniques used to detect cycle are BFS or DFS cycle detection algorithm.

**Q24: Find out the Kth smallest element in an unsorted array?**

Ans-

```
 public int kthElement(int[] array , int k){
Arrays.sort(array);
return array[k-1];
}
```

**Q25: How to find the shortest path between two vertices?**

Ans- Shortest path between two vertices is determined by using a shortest path algorithm called Dijkstra's Algorithm.
Dijkstra's Algorithm is also called single source shortest path algorithm.