1) What do you mean by a Data structure?
-->Data structures are the ways of organizing and storing data in a computer so that we can perform
several operations efficiently on it. It is widely used in every aspect of computer science.
Some examples of Data Structures are arrays, Linked List, Stack, Queue, etc.

2) What are some of the applications of DS?
-->1.Stacks are used in keeping track of method calls in a program.
2.Queues are used for interprocess communications.
3.Binary Search Trees are used for implementing maps and sets.
4.B-Trees are used in Database designs
5.Trees are used in File-System.

3) What are the advantages of a Linked list over an array?
-->Linked List allocates the memory dynamically unlike array who has fixed size. All the nodes of
linked list are non-contiguously stored in the memory and linked together with the help of pointers.
Sizing is no longer a problem since we do not need to define its size at the time of declaration.
List grows as per the program's demand and limited to the available memory space.

4) Write the syntax in C to create a node in the singly linked list.

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

5) What is the use of a doubly-linked list when compared to that of a singly
linked list?
-->In a singly linked list, we could traverse only in one direction, because each node contains address of the next
node and it doesn't have any record of its previous nodes. However, doubly linked list overcome this limitation of
singly linked list. Due to the fact that, each node of the list contains the address of its previous node, we can
find all thedetails about the previous node as well by using the previous address stored inside the previous part
of each node.

6) What is the difference between an Array and Stack?
-->Stack is a ordered collection of items.Stack is a dynamic object whose size is constantly changing as items are pu
shed
and popped .Stack may contain different data types.
while,Array is an ordered collection of items.Array is a static object the no of items are fixed and is assigned while
declaring of the array.It contains same data types.

7) What are the minimum number of Queues needed to implement the priority queue?
--> The answer is two.One queue is used for the actual storing of data, and the other one is used for storing the priori
ties.

8) What are the different types of traversal techniques in a tree?
-->1.preorder
In a preorder traversal, we visit the root node first, then recursively do a preorder traversal of the left subtree, follo
wed
by a recursive preorder traversal of the right subtree.
2.Inorder
In an inorder traversal, we recursively do an inorder traversal on the left subtree, visit the root node, and finally do
a
recursive inorder traversal of the right subtree.
3.postorder

In a postorder traversal, we recursively do a postorder traversal of the left subtree and the right subtree followed by a
visit to the root node.

9) Why it is said that searching a node in a binary search tree is efficient than that of a simple binary tree?
-->Binary Search Tree is sorted thats why searching is more efficient than simple binary tree.

10) What are the applications of Graph DS?
-->1.Facebook: Each user is represented as a vertex and two people are friends when there is an edge between two v ertices.
Similarly friend suggestion also uses graph theory concept.
2.Google Maps: Various locations are represented as vertices and the roads are represented as edges and graph the ory is used
to find shortest path between two nodes.
3.Recommendations on e-commerce websites: The “Recommendations for you” section on various e-commerce w ebsites uses graph theory
to recommend items of similar type to user’s choice.

11) Can we apply Binary search algorithm to a sorted Linked list?
-->The binary search algorithm cannot be applied to sorted linked list because in linked list we do not have the indic es to access items
in order to do so you have traverse the list each time.

12) When can you tell that a Memory Leak will occur?
-->A memory leak is a situation where a program allocates memory to do an operation and when the operation is co mpleted, the memory is not
de-allocated i.e.returned back to the pool for re-use later on. If the operation is frequent, the more the program runs, the more memory
it allocates, which can have a negative impact on the operation of the entire system.

13) How will you check if a given Binary Tree is a Binary Search Tree or not?
-->A binary search tree has following properties:
1.The left subtree of a node contains only nodes with keys less than the node’s key.
2.The right subtree of a node contains only nodes with keys greater than the node’s key.
3.Both the left and right subtrees must also be binary search trees.

14) Which data structure is ideal to perform recursion operation and why?
-->Stack is used to perform recursion operation.Beacuse in recursion we call fuction during each operation
and the output producedby that fuction call is passed to next. This matches the requirement of a LIFO data
structure hence we use Stack.

15) What are some of the most important applications of a Stack?
-->Expression Evaluation and Conversion, Backtracking,Function call,Parenthesis Checking,String reversal etc.

16) Convert the below given expression to its equivalent Prefix And Postfix notations.
17)Sorting a stack using a temporary stack
-->
```
    public Stack<Integer> sort(Stack<Integer> stack) {

        Stack<Integer> tempStack = new Stack<>();

        while(!stack.isEmpty()) {
            int x = original.pop();
            while(!tempStack.isEmpty() && tempStack.peek() > x) {
               stack.push(tempStack.pop());
            }
            tempStack.push(x);
        }
        return tempStack;
    }

    public static void main(String args[]) {

        StackSorting s = new StackSorting();
        Stack<Integer> stack = new Stack<>();
        stack.add(8);
        stack.add(9);
        stack.add(7);
        stack.add(12);
        System.out.println(stack);
        Stack<Integer> sortedStack= s.sort(stack);
        System.out.println(sortedStack);
    }
}
```

18)Program to reverse a queue
-->

```
 public class reverseQueue{

    static void show()
    {
        while (!queue.isEmpty()) {
            System.out.print( queue.peek());
            queue.remove();
        }
 }
    static void reversequeue()
    {
        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<Integer> qL= new LinkedList<Integer>();
        qL.add(10);
        qL.add(20);
        qL.add(30);
        qL.add(40);
        qL.add(50);
        reversequeue();
        show();
    }
```

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

```
public class Reverse{

    static Queue<Integer> queue;
    static void revKelments(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 show() {
        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 = 4;
        revKelments(k);
        show();
    }
}
```

20)Program to return the nth node from the end in a linked list
21)Reverse a linked list
-->public class myList{

```
    public static void main(String a[]){

        LinkedList<String> ls = new LinkedList<String>();
        ls.add("Abhi");
        ls.add("Beast");
        ls.add("Dave");
        ls.add("Donno");
        ls.add("Link");
        Collections.reverse(ls);
        for(String str: ls){
            System.out.println(str);
        }
    }
}
```

22)Replace each element of the array by its rank in the array

```
-->class Arr{
    public static void rankArr(int[] arr)
    {
        Map<Integer, Integer> map = new TreeMap<>();

        for (int j = 0; j < arr.length; j++) {
            map.put(arr[j], j);
        }

        int rank = 1;

        for (Map.Ent<Integer, Integer> e : map.entSet()) {
            arr[e.getValue()] = rank++;
        }
    }

    public static void main(String[] args)
    {
        int[] A = { 15, 22, 8, 12, 4, 20, 8 };
        rankArr(A);
        System.out.println(Arrays.toString(A));
    }
}
```

23) Check if a given graph is a tree or not
--> Graph is tree if:
1. It has number of edges one less than number of vertices. i.e. E=n-1
2. Graph is connected.
3. There are no cycles.

24) Find out the Kth smallest element in an unsorted array
-->

```
class GFG
{
    public static int kSmall(int arr[],int k)
    {
        Arrays.sort(arr);
        return arr[k-1];
    }
    public static void main(String[] args)
    {
        int arr[] = new int[]{8, 7, 4, 12, 56,16};
        int k = 3;
        System.out.print(kSmall(arr, k) );
    }
}
```

25) How to find the shortest path between two vertices
-->Using Dijkstra's Algorithm we can find shortest path between two vertices in a graph.Dijkstra’s algorithm which updates the shortest path between the current node and all of its neighbors.After updating the distance of all of the neighbors it moves to the node with the lowest distance and repeats the process with all unvisited neighbors. This process continues until the entire graph has been visited.