1) **What do you mean by a Data structure?**
   A data structure is a particular way of organizing data in a computer so that it can be used effectively.

2) **What are some of the applications of DS?**

1.Array can be used for sorting elements, can perform matrix operation & can be used in CPU scheduling.
2.Stack is used in Expression evaluation, Forward and backward feature in web browsers, syntax parsing, Used in many algorithms like Tower of Hanoi, histogram problem etc.
3.Queue is used when a resource is shared among multiple consumers like in CPU scheduling, Disk Scheduling. It is also used in Palindrome recognition.
4.Tree is used as dictionary, such as one found on a mobile telephone for autocompletion and spell-checking.
5.Hash Table is used for fast data lookup - symbol table for compilers, database indexing, caches, Unique data representation.

6.Processor speed: To handle very large amout of data, high speed processing is required, but as the data is growing day by day to the billions of files per entity, processor may fail to deal with that much amount of data.

7.Data Search: Consider an inventory size of 106 items in a store, If our application needs to search for a particular item, it needs to traverse 106 items every time, results in slowing down the search process.

3) **What are the advantages of a Linked list over an array?**

1.Dynamic in size (can grow and shrink anytime)

2.in O(1) time complexity insertion and deletion

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 singly linked list traversal is possible in only forward direction whereas in doubly-linked list it is possible in both forward and backward direction. Deletion and reverse the list is easy is doubly linked list as compared to singly linked list.

6) **What is the difference between an Array and Stack?**

In an array, you have a list of elements and you can access any of them at any time. But in a stack, there's no random-access operation; there are only Push, Peek and Pop, all of which deal exclusively with the element on the top of the stack. A stack is data-structure which has a last in first out policy.

7) **What are the minimum number of Queues needed to implement the priority queue?**

2 queues are used. One for sorting and second one for priorities.

8) **What are the different types of traversal techniques in a tree?**
Preorder traversal,postorder traversal, Inoder traversal

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

Binary tree is unordered hence slower in process of insertion, deletion and searching. Searching of an element is faster in Binary search tree than binary tree due to the ordered characteristics. In binary search tree the left subtree has elements less than the nodes element and the right subtree has elements greater than the nodes element.

10) **What are the applications of Graph DS?**

➔ Used to represent the flow pf computation.

➔ In Operating System, we come across the Resource Allocation Graph where each process and resources are considered to be vertices. Edges are drawn from resources to the allocated process, or from requesting process to the requested resource. If this leads to any formation of a cycle then a deadlock will occur.

➔ In World Wide Web, web pages are considered to be the vertices. There is an edge from a page to other page.

11) **Can we apply Binary search algorithm to a sorted Linked list?**

Binary search can not be applied on sorted linked list. It works on indexes.

12) **When can you tell that a Memory Leak will occur?**
memory leak is a type of resource leak that occurs when a computer program incorrectly manages memory allocations in a way that memory which is no longer needed is not released. A memory leak may also happen when an object is stored in memory but cannot be accessed by the running code.

13)How will you check if a given Binary Tree is a Binary Search Tree or not?

The left subtree of a node contains only nodes with keys less than the node's key, The right subtree of a node contains only nodes with keys greater than the node's key and 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 has the LIFO 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.

15) What are some of the most important applications of a Stack?

Expression handling: ex- infix to postfix or infix to prefix, To check parenthesis matching, Backtracking procedure: ex-N-Queen Problem and memory management.

16) Convert the below given expression to its equivalent Prefix and Postfix notations.

*+AB-CD (Infix : (A+B) * (C-D)) => AB+CD-* (Infix : (A+B * (C-D))

17)Sorting a stack using a temporary stack

```
#include <bits/stdc++.h>
using namespace std;

stack<int> sortStack(stack<int> &input)
{
    stack<int> tmpStack;

    while (!input.empty())
    {
        int tmp = input.top();
        input.pop();

        while (!tmpStack.empty() && tmpStack.top() > tmp)
        {
            input.push(tmpStack.top());
            tmpStack.pop();
        }
        tmpStack.push(tmp);
    }

    return tmpStack;
}

int main()
{
    stack<int> input;
    input.push(1);
    input.push(2);
    input.push(3);
    input.push(4);
```

```
        input.push(5);
        input.push(6);

        stack<int> tmpStack = sortStack(input);
        cout << "Sorted Stack is :\n";

        while (!tmpStack.empty())
        {
                cout << tmpStack.top()<< " ";
                tmpStack.pop();
        }
}
```

18)Program to reverse a queue

```
#include <bits/stdc++.h>
using namespace std;

void Print(queue<int>& Queue)
{
        while (!Queue.empty()) {
                cout << Queue.front() << " ";
                Queue.pop();
        }
}

void reverseQueue(queue<int>& Queue)
{
        stack<int> Stack;
        while (!Queue.empty()) {
                Stack.push(Queue.front());
                Queue.pop();
        }
```

```
	while (!Stack.empty()) {
		Queue.push(Stack.top());
		Stack.pop();
	}
}

int main()
{
	queue<int> Queue;
	Queue.push(1);
	Queue.push(2);
	Queue.push(3);
	Queue.push(4);

	reverseQueue(Queue);
	Print(Queue);
}
```

19) Program to reverse first k elements of a queue

```
#include <bits/stdc++.h>
using namespace std;

void reverseQueueFirstKElements(
	int k, queue<int>& Queue)
{
	if (Queue.empty() == true
		|| k > Queue.size())
		return;
	if (k <= 0)
		return;

	stack<int> Stack;
```

```
        for (int i = 0; i < k; i++) {
                Stack.push(Queue.front());
                Queue.pop();
        }

        while (!Stack.empty()) {
                Queue.push(Stack.top());
                Stack.pop();
        }

        for (int i = 0; i < Queue.size() - k; i++) {
                Queue.push(Queue.front());
                Queue.pop();
        }
}

void Print(queue<int>& Queue)
{
        while (!Queue.empty()) {
                cout << Queue.front() << " ";
                Queue.pop();
        }
}

int main()
{
        queue<int> Queue;
        Queue.push(10);
        Queue.push(20);
        Queue.push(30);
        Queue.push(40);
        int k = 2;
        reverseQueueFirstKElements(k, Queue);
        Print(Queue);
}
```

20)Program to return the nth node from the end in a linked list

```
#include<bits/stdc++.h>
using namespace std;

struct Node
{
int data;
struct Node* next;
};

void printNthFromLast(struct Node *head, int n)
{
struct Node *main_ptr = head;
struct Node *ref_ptr = head;

int count = 0;
if(head != NULL)
{
        while( count < n )
        {
                if(ref_ptr == NULL)
                {
                printf("%d is greater than the no. of "
                                        "nodes in list", n);
                return;
                }
```

```
                    ref_ptr = ref_ptr->next;

                    count++;

          }


          while(ref_ptr != NULL)

          {

                    main_ptr = main_ptr->next;

                    ref_ptr = ref_ptr->next;

          }

          printf("Node no. %d from last is %d ",

                              n, main_ptr->data);

}

}


void push(struct Node** head_ref, int new_data)

{

struct Node* new_node = new Node();


new_node->data = new_data;


new_node->next = (*head_ref);


(*head_ref) = new_node;


}


int main()

{

struct Node* head = NULL;

push(&head, 1);

push(&head, 2);
push(&head, 3);
push(&head, 4);
printNthFromLast(head, 2);
}
```

21)Reverse a linked list

```
#include <bits/stdc++.h>
using namespace std;

struct Node {
        int data;
        struct Node* next;
        Node(int data)
        {
                this->data = data;
                next = NULL;
        }
};

struct LinkedList {
        Node* head;
        LinkedList()
        {
                head = NULL;
        }

        Node* reverse(Node* head)
        {
                if (head == NULL || head->next == NULL)
                        return head;

                Node* rest = reverse(head->next);
                head->next->next = head;

                head->next = NULL;

                return rest;
        }

        void print()
        {
                struct Node* temp = head;
                while (temp != NULL) {
                        cout << temp->data << " ";
                        temp = temp->next;
                }
        }

        void push(int data)
        {
                Node* temp = new Node(data);
                temp->next = head;
                head = temp;
        }
};

int main()
{
        LinkedList an;
        an.push(1);
        an.push(2);
        an.push(3);
        an.push(4);
        cout << "Entered linked list\n";
        an.print();

        an.head = an.reverse(an.head);

        cout <<"After reversing the list is "<<endl;
        an.print();
        return 0;
}
```

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

```
 import java.util.*;

 class RankReplace {

        static void changeArr(int[] input)
        {
                int newArray[] = Arrays.copyOfRange(input,0,input.length);

                Arrays.sort(newArray);
                int i,j;
                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 = { 100, 2, 70, 2 };
            changeArr(arr);
            System.out.println(Arrays.toString(arr));
        }
}
```

23) Check if a given graph is a tree or not

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

```
#include <bits/stdc++.h>
using namespace std;

int findkthSmallest(int arr[], int n, int k)
{
   sort(arr, arr + n);
   return arr[k - 1];
}

int main()
{
   int arr[] = { 11,5,85,65,0,5 };
   int n = sizeof(arr) / sizeof(arr[0]);
        int k = 4;
```

```
    cout << k<<"smallest element is:  " << findkthSmallest(arr, n, k);
    return 0;
}
```