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

In simple words, Data structure means if you have collection of data how you should organize it so that you can perform basic data manipulation, search, sort etc. operations as quick as possible in others words effectively. Data Structures is all about having an organised way of elements which are used in all computer and web applications.

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

**Applications of DS are:-**

* **Arrays:** Implementation of other data structures, Execution of matrices and vectors, Dynamic memory allocation, Pointer container, Control tables.
* **Stack:** Evaluation of expressions, Backtracking, Runtime memory management, Arrangement of books in a library.
* **Queue:** Here, the data sent need not be received at the same rate at which it was sent. A certain system resource is to be shared between different processes.
* **Linked-List:** Representation of sparse matrices, Non-contiguous data storage, Implementation of non-binary tree or other data structures, Dynamic memory management, Equalizing parenthesis, Symbol tables.
* **Graph:** Computer networking, Problem solutions involving 'Depth-First' search or 'Breadth-First' search algorithms, Representation of matrices, Study of molecular interactions in Chemistry.
* **Tree:** Representation of data lists, quickly accessible data storage, Representation of hierarchal data, Routing of algorithms.
* **Set:** Mapping of data, Common data storage

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

**Arrays:** Arrays have Fixed Size. Operations of Insertion and deletion are difficult and they also have lots of Memory wastage.

**Linked List:** Linked list has Dynamic size with ease of Insertion and deletion and no memory wastage either.

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

Struct node

{

int data;

Struct node \*next;

};

**Q5. 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 only in forward direction whereas the doubly linked list can be traversed both in forward and backward direction. It is more efficient in the delete and reverse operation whenever the pointer to node is given in doubly linked list.

**Q6. What is the difference between an Array and Stack?**

|  |  |
| --- | --- |
| **Array** | **Stack** |
| 1. 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 | 1. Stacks are usually based on the LIFO principle, i.e., the element inserted at the last, is the first element to come out of the list. |
| 2. Insertion and deletion in array can be done at any index in the array. | 2. Operations on Insertion and deletion in stacks take place only from one end of the list called the top. |
| 3. Array has a fixed size. | 3. Stack has fixed size. |
| 4. The array can contain elements of the same data types. | 4. Stack can contain elements of different data type. |
| 5. Different types of Arrays are 1D, 2D, etc | 5. Stack has only one type. |

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

If you use predefined queues from STL or utility package then we need two queues one for storing data and other for its priority. If we program queue we can add another variable in the node like priority with int type.

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

* Preorder Traversal,
* Inorder Traversal
* Postorder Traversal

**Q9. 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 elements and the right subtree has elements greater than the nodes element.

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

* In Computer science graphs are used to represent the flow of computation.
* In Facebook, users are considered to be the vertices and if they are friends then there is an edge running between them. Facebook’s Friend suggestion algorithm uses graph theory. Facebook is an example of undirected graph.
* In World Wide Web, web pages are considered to be the vertices.

**Q11. Can we apply Binary search algorithm to a sorted Linked list?**

Yes, Binary search is possible on the linked list if the list is ordered and you know the count of elements in 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.

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

Memory leak occurs when we create a memory in the heap and forget to delete it after using it. It may also occur 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.

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

Binary search trees have following properties:

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

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

Stack has the 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.

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

Applications of stack:

* Expression handling
* To check parenthesis matching.
* Backtracking procedure
* Memory Management.

**Q16. Convert the below given expression to its equivalent Prefix And Postfix notations**

Question is incomplete.

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

* Create Stack s and temp, int min;
* Add elements in stack s;
* Now for i=0 ; i< Stack s length ; i++
* For j=i; j< Stack s length ; j++
* Element = pop Stack s
* Check the element with min and find min
* Push the element into temp
* Repeat step 4
* Now push the min into Stack s
* Push all the elements of temp into Stack s except min
* Repeat step 3
* Stack is sorted, end

**Q18. Program to reverse a queue**

Program to reverse a queue

Queue q;

Stack s;

Int len=q1.length(), temp;

For(int i=0;i<q1.size();i++)

{

s.push(q.poll());

}

For(int i=0;i<s1.size();i++)

{

q.add(s.poll());

}

**Q19. Program to reverse first k elements of a queue?**

Queue q, q1;

Stack s;

Int k;

For(int i=0;i<k;i++)

{

s.push(q.poll());

}

For(int i=0;i<s1.size();i++)

{

q1.add(s.poll());

}

For(int i=0;i<s1.size()-k;i++)

{

q1.add(q.poll());

}

q=q1;

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

LinkedList l;

Int len,value;

Iterator I = l.iterator();

While(I.hasNext())

{

Len++;

I.next();

}

I = l.iterator();

For(int i=0;i< len-n ;i++)

{

Value = I.next();

}

Return value;

**Q21. Reverse a linked list**

    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;

    }

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

class Demo{

      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]);

        }

    }

**Q23. Check if a given graph is a tree or not**

int V;

LinkedList<Integer> adj[];

    Boolean isCyclic(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;

 }

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

 public static int kthSmallest(Integer[] arr, int k)

    {

        Arrays.sort(arr);

   return arr[k - 1];

    }

**Q25. How to find the shortest path between two vertices**

* Input the graph.
* Input the source and destination nodes.
* Find the paths between the source and the destination nodes.
* Find the number of edges in all the paths and return the path having the minimum number of edges.
* In this way we will get the shortest path between the two vertices. For optimal solution use Dijkstra's Algorithm.