Guess solution

Q1: State the difference between static and dynamic data structures.

Ans: Static Data structure has fixed memory size whereas in Dynamic Data Structure, the size can be randomly updated during run time which may be considered efficient with respect to memory complexity of the code. Static Data Structure provides more easier access to elements with respect to dynamic data structure. Unlike static data structures, dynamic data structures are flexible.

Q2: Why we use stacks as data structure?

Ans: We use stack data structure where we need last entered data. Stack work on LIFO (last in fist out). Whenever we use recursion, the computer uses stack to execute the program.

Q3: State the benefits of quick sort.

- It is faster than merge sort.
- It uses recursion method (divide and conquer) to sort
- it uses to sort large list in less time
- average case is (n log n)

Q4: What is meant by asymptotic notations?

Ans: In asymptotic notation we analyze a program or algorithm is such a way to know that it runs in best time or worst time. Time is used as a parameter and this analysis perform before execution.

Q5: How breath first search traversal works?

Ans: It enters the starting vertex to queue then it enqueue its adjacent unvisited vertex to queue and makes it visited and dequeue it from queue. If no adjacent vertex is present it dequeue it from vertex. Now new vertex is at queue's head it preforms the same operation till no adjacent unvisited vertex is present.

It uses ordered traversal means it traverse from all elements in a level then it goes to lower level when it performs on tree.

Q6: Describe ADT?

Ans: ADT stand for Abstract data type is special data type that defines types of data it stores as well as methods or operations performed on it. It is also known as non-primitive data types. Arrays, stack, linked list and queues are the example of ADT.

Q7: List down linear and non-linear data structures.

Ans: **Linear data structure** (stack, queue, linked-list, array)

Nonlinear data structure (tree and map)

Q8: State the benefits of stacks over queues.

Ans: The benefit of stack over queue is that it will return the last input data.

Q9: What is two-dimensional Array?

Ans: The two-dimensional array can be defined as an array of arrays. The 2D array is organized as matrices which can be represented as the collection of rows and columns.

In java 2D array of size 3x3 (rows X colums)

```
int[][] arr = int[3][3];
```

Q10: Define term efficiency in data structures.

Ans: The term efficiency in data structure means less uses of resources such as time and space in order of performing retrieval and storage of data.

Q11: Define Hashing?

Ans: Hashing is a technique or process of mapping keys, values into the hash table by using a hash function. It is done for faster access to elements.

Q12: Why we use sorting in data structures?

Ans: Accessed data in sorted array is much easier and faster than unsorted array therefore we use sorting in data structures

Q13: Why we use queues as data structure?

Ans: Queues uses FIFO (fist in fast out) technique to retrieval data. So, where we need data in sequence of input, we use queues.

Q14: What is meant by greedy algorithms?

Ans: A greedy algorithm is any algorithm that follows the problem-solving heuristic of making the locally optimal choice at each stage.

Q15: What is meant by time complexity of an algorithm?

Ans: time complexity of an algorithm means time consumed to run an algorithm. But time consumption is different on different inputs so

There are 3 cases in which an algorithm runs

1. best case (min time) Big Omega
2. worst case (max time) Big oh
3. average case big theta

Q16: State the difference between primitive and non-primitive data types.

Primitive	Non-Primitive
Not need to create object.	Need to create an object
Already defines in java	Defined by user
Define data	Define data as well as method perform on that data
Store only defined data within defined range	It can store data of many data types within user defined range
Byte, short, int, float, double, long, Boolean, char	Arrays, string, classes, objects

Q17: What is meant by reference types?

Ans: In Java, **non-primitive** data types are known as **reference types**. In other words, a variable of class type is called **reference data type**. It contains the address (or reference) of dynamically created objects. For example, if Demo is a class and we have created its object **d**, then the variable d is known as a reference type.

Q18: What is minimum spanning tree?

Ans: The minimum spanning tree is a connected subgraph of G that spans all vertices at minimum cost.

==========

Long question

Q.1 If you have to choose between array and link list in your program what do you choose.

Justify your answer with logical reasoning.

Ans: I choose Linked List for my program because

- I can insert new node at fist, at last, or anyware in a list easily without moves data to next indexes. we just link new node to next and previous nodes. it will save time.

- I can also delete at fist, at last, or from anyware in a linked list very easily and deletion from a linked lisk not makes any holes like deletion in array and not need to move data to backword indexes after delition. we just link one step previous and one step next node. it will save time

- linked list provide us Dynamic size.

- unlike arrays it provide methods to show linked list.

Q2: Write the code in JAVA to implement Double linked list in your program.
Ans:

```
public class ExecuteProgram {
	public static void main(String[] args) {
		LinkedListDoubly l1 = new LinkedListDoubly();
		l1.insert(1);
		l1.insert(2);
		l1.insert(3);
		l1.insert(4);
		l1.insert(5);
		l1.delete();
		l1.deleteAtFist();
		l1.insertAtFist(1);
		l1.show();
	}
```

```
}


public class LinkedListDoubly {

	Node head = null;
	Node tail = null;

	class Node{
		Node pre;
		Node next;
		int data;

		Node(int data){
			pre = next = null;
			this.data = data;
		}
}

public boolean isEmpty() {
	if(head == null) {
		return true;
	}else {
		return false;
	}
}

public void makeEmpty() {
	if(isEmpty()) {
		System.out.println("already empty!");
	}else {
		head = tail = null;
	}
}

public void insert(int data) {
	if(isEmpty()) {
		head = tail = new Node(data);
	}else {
		Node node = new Node(data);
		node.pre = tail;
		tail.next = node;
		tail = node;
	}
```

```
}

public void insertAtFist(int data) {
	if(isEmpty()) {
		head = tail = new Node(data);
	}else {
		Node node = new Node(data);
		node.next = head;
		head.pre = node;
		head = node;
	}
}
public void delete() {
	if(isEmpty()) {
		System.out.println("already empty!");
	}else if(head == tail) {
		head = tail = null;
	}else {
		tail = tail.pre;
		tail.next =null;
	}
}
public void deleteAtFist() {
	if(isEmpty()) {
		System.out.println("already empty!");
	}else {
		head = head.next;
		head.pre =null;
	}
}
public void show() {
	if (isEmpty()) {
		System.out.println("already empty!");
	} else {
		Node node = head;
		while(node != null) {
			System.out.println(node.data);
			node = node.next;
		}
	}
	}
}
```

Q.3 Your fellow mate has requested you to implement the INSERTION SORT functionality on a user-defined array.

Ans:

```
public class ExecuteProgram {
	public static void main(String[] args) {
		display(InsertionSort.sort(input()));
	}


	public static void display(int[] arr) {
		for(int i=0;i<arr.length;i++) {
			System.out.println(i+" : "+arr[i]);
		}
	}
public static int[] input() {
		Scanner input=new Scanner(System.in);
		System.out.println("Enter size of array");
		int n=input.nextInt();
		int[] arr= new int[n];
		for(int i=0;i<n;i++) {
			System.out.println("Enter "+i+" index value of array");
			arr[i]=input.nextInt();
		}
		return arr;
	}
}

public class InsertionSort {
	public static int[] sort(int[] arr) {
		for(int i=1; i<arr.length; i++) {
			int j = i;
			int select = arr[i];

			while(j>0 && arr[j-1]>select) {
				arr[j] = arr[j-1];
				j--;
			}
			arr[j] = select;
		}
		return arr;
	}
}
```

Q.4 What is binary search tree? What is the possible maximum number of nodes in BST of depth d. Make a BST for the following sequence of numbers and traverse all types of traversal.

45,32,90,21,78,77,81,132,90,96,25,110,36

Ans: A Binary Search tree (BST) can be defined as a type of binary trees, in which the nodes are arranged in a specific order. also called as ordered or sorted binary tree.

=> Properties of a binary tree

1) In a binary search tree, the value of all the nodes in the left sub-tree is less than the value of the root.

2) Similarly, value of all the nodes in the right sub-tree is greater than or equal to the value of the root.

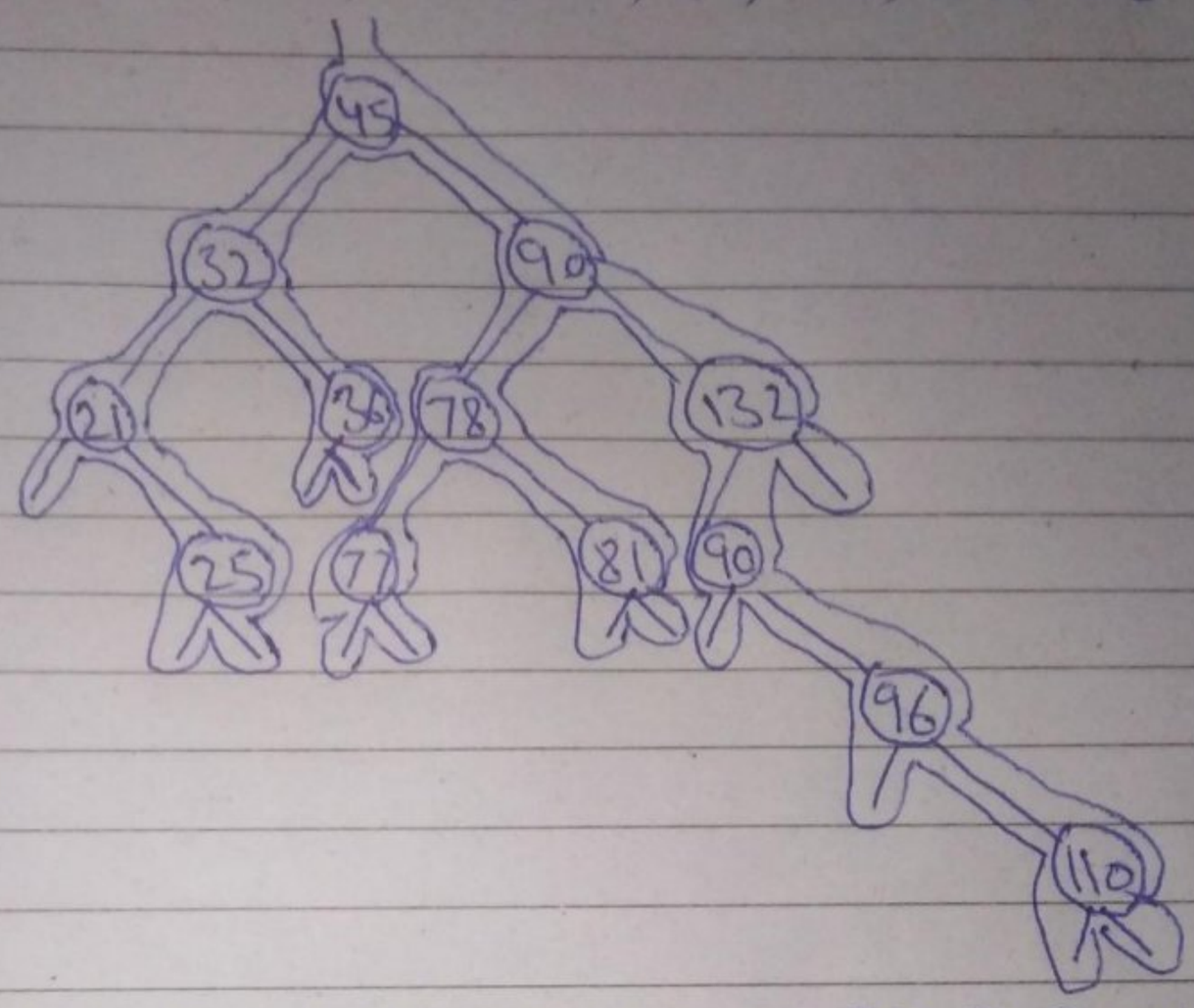
3) This rule will be recursively applied to all the left and right sub-trees of the root.

(make an image here)

The possible maximum number of nodes in BST of depth d is defined by the formula n = (2^d+1)-1

Date ...../...../.....

45, 32, 90, 21, 78, 77, 81, 132, 90, 96, 25, 110, 36



Level order (45, 32, 90, 21, 36, 78, 132, 25, 77, 81, 90, 96, 110)

Pre Order (45, 32, 21, 25, 36, [illegible], 90, 78, 77, 81, 132, 90, 96, 110)

In order (21, 25, 32, 36, 45, 77, 78, 81, 90, 90, 96, 110, 132,)

Post order (25, 21, 36, 32, 77, 81, 78, [illegible] 110, 96, 90, 132, 90, 45)

Q.5. Consider a scenario, where user has entered randomly into an array of user-defined size. Now he/she want to sort all the input in ascending form. The user has directed you to sort the number using SELECTON SORT.

Ans:

```
import java.util.Scanner;

public class ExecuteProgram {

	public static void main(String[] args) {
		display(SelectionSort.sort(input()));
	}



public static void display(int[] arr) {
		for(int i=0;i<arr.length;i++) {
			System.out.println(i+" : "+arr[i]);
		}
}



public static int[] input() {
		Scanner input=new Scanner(System.in);
		System.out.println("Enter size of array");
		int n=input.nextInt();
		int[] arr= new int[n];
		for(int i=0;i<n;i++) {
			System.out.println("Enter "+i+" index value of array");
			arr[i]=input.nextInt();
```

```
                }
                return arr;
        }
}
```

```
public class SelectionSort {
        public static int[] sort(int[] arr) {
for(int i=0; i<arr.length; i++) {
        int select = i;
        for(int j=i+1; j<arr.length ; j++) {
                if(arr[select]>arr[j]) {
                        select = j;
                }
        }
        int temp = arr[i];
        arr[i]=arr[select];
        arr[select]=temp;
}
return arr;
        }
}
```

Q.8 Imagine a class of 20 students of Subject A. Teacher has entered randomly marks of students. Now teacher wants to know who got the 2nd and 3rd highest marks in the class.

Implement the above scenario in array of size 20.

Ans:

```
import java.util.Scanner;

public class ExecuteProgram {
	public static void main(String[
] args) {
		Students[] studentArr = new Students[20];
		for(int i=1; i<=20; i++) {
			System.out.println("enter name and marks of student "+i);
			Scanner input = new Scanner(System.in);
			String name = input.next();
			int marks = input.nextInt();
			studentArr[0] = new Students(name, marks);
		}
		Students.sortByMarks(studentArr);

			System.out.println("2nd highest
marks\n"+studentArr[2].name+":"+studentArr[2].marks);
			System.out.println("3nd highest
marks\n"+studentArr[3].name+":"+studentArr[3].marks);
	}
}

public class Students {
	public String name;
```

```
public int marks;
public String subject;

public Students(String name,int marks) {
        this.name = name;
        this.marks = marks;
        this.subject = "A";
}
    public static void sortByMarks(Students[] arr) {
        for(int i=0; i<arr.length-1; i++) {
            for(int j=0; j<arr.length-1-i; j++) {
                if(arr[j].marks > arr[j+1].marks) {
                    Students temp = arr[j];
                    arr[j] = arr[j+1];
                    arr[j+1] = temp;
                }
            }
        }
    }
}
```

Q.10 Describe the advantages of BST over Hashing. Also write a program to insert or delete an item from circular queue.

Ans:

1) We can get all keys in sorted order by just doing Inorder Traversal of BST. This is not a natural operation in Hash Tables and requires extra efforts.

2) Doing order statistics, finding closest lower and greater elements, doing range queries are easy to do with BSTs. Like sorting, these operations are not a natural operation with Hash Tables.

3) BSTs are easy to implement compared to hashing, we can easily implement our own customized BST. To implement Hashing, we generally rely on libraries provided by programming languages.

4) With Self-Balancing BSTs, all operations are guaranteed to work in O(Logn) time. But with Hashing, Θ(1) is average time and some particular operations may be costly, especially when table resizing happens.

```
public class Queue {

        int[] arr;
        int front;
        int rear;
        int size;
        int capacity;

        public Queue(int capacity) {
                arr = new int[capacity];
                this.capacity = capacity;
```

```
            front = size = 0;
            rear = capacity-1;
}

public boolean isEmpty() {
            return size == 0;
}

public boolean isFull() {
            return size == capacity;
}

public void enQueue(int item) {
            if (this.isFull()) {
                        System.out.println("queue is full");
            } else {
                        rear = (rear+1) % capacity;
                        arr[rear] = item;
                        size++;
            }
}

public int deQueue() {
            if (this.isEmpty()) {
                        System.out.println("queue is empty");
                        return Integer.MIN_VALUE;
            } else {
                        int item = arr[front];
                        front = (front + 1) % capacity;
```

```
                size--;
                return item;
        }
}

public int front() {
        if (this.isEmpty()) {
                System.out.println("queue is empty");
                return Integer.MIN_VALUE;
        } else {
                return arr[front];
        }
}

public int rear() {
        if (this.isEmpty()) {
                System.out.println("queue is empty");
                return Integer.MIN_VALUE;
        } else {
                return arr[rear];
        }
}

public void show() {
        if (this.isEmpty()) {
                System.out.println("queue is empty");
        } else {
                for(int i=0; i<arr.length; i++) {
                        System.out.println(arr[i]);
```

```
                    }
                }
            }
}
```

Q.10 Describe the advantages of BST over Hashing. Also write a program to insert or delete an item from circular queue.

Ans:

1) We can get all keys in sorted order by just doing Inorder Traversal of BST. This is not a natural operation in Hash Tables and requires extra efforts.

2) Doing order statistics, finding closest lower and greater elements, doing range queries are easy to do with BSTs. Like sorting, these operations are not a natural operation with Hash Tables.

3) BSTs are easy to implement compared to hashing, we can easily implement our own customized BST. To implement Hashing, we generally rely on libraries provided by programming languages.

4) With Self-Balancing BSTs, all operations are guaranteed to work in O(Logn) time. But with Hashing, Θ(1) is average time and some particular operations may be costly, especially when table resizing happens.

```
public class Queue {
        int[] arr;
        int front;
        int rear;
        int size;
        int capacity;
```

```
public Queue(int capacity) {
        arr = new int[capacity];
        this.capacity = capacity;
        front = size = 0;
        rear = capacity-1;
}

public boolean isEmpty() {
        return size == 0;
}

public boolean isFull() {
        return size == capacity;
}

public void enQueue(int item) {
        if (this.isFull()) {
                System.out.println("queue is full");
        } else {
                rear = (rear+1) % capacity;
                arr[rear] = item;
                size++;
        }
}

public int deQueue() {
        if (this.isEmpty()) {
                System.out.println("queue is empty");
                return Integer.MIN_VALUE;
```

```
		} else {
			int item = arr[front];
			front = (front + 1) % capacity;
			size--;
			return item;
		}
	}

	public int front() {
		if (this.isEmpty()) {
			System.out.println("queue is empty");
			return Integer.MIN_VALUE;
		} else {
			return arr[front];
		}
	}

	public int rear() {
		if (this.isEmpty()) {
			System.out.println("queue is empty");
			return Integer.MIN_VALUE;
		} else {
			return arr[rear];
		}
	}

	public void show() {
		if (this.isEmpty()) {
			System.out.println("queue is empty");
```

```
                } else {
                        for(int i=0; i<arr.length; i++) {
                                System.out.println(arr[i]);
                        }
                }
        }
}
```

Written by

Muhammad Abdullah Madni

Arranged by

Muhammad Ramzan