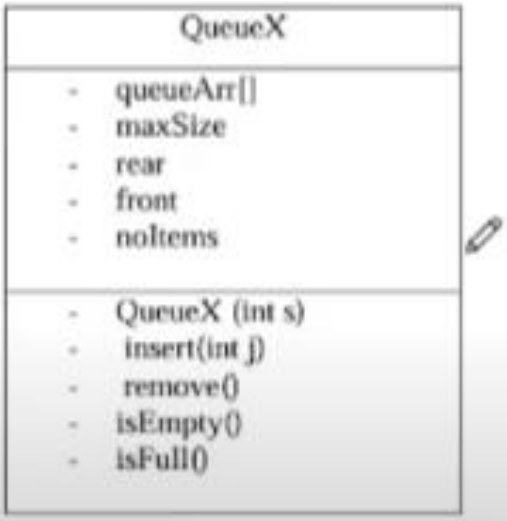
Question

i) You are required to build the following QueueX class in your program.



ii) Write a main program to create an object called *mainQueue* with 5 elements of the QueueX class. This is used to store transactions IDs

iii) Allow the user to input 5 transaction IDs from the keyboard and store them in *printerQueue*.

```
Enter transaction ID 1: 145
Enter transaction ID 2: 666
Enter transaction ID 3: 112
Enter transaction ID 4: 598
Enter transaction ID 5: 123
```

iv) You are required to send these transactions to separate PCs based on the transaction ID. Transactions sent to PC1 contains even transaction IDs and transactions sent to PC2 contain odd IDs. Create two objects called *evenQueue* and *oddQueue* to store these details.

```
(Eg: ID 666 is sent to PC1 and ID 123 is sent to  PC2 )
```

v) Write the code to remove the numbers and display the result as follows.

```java
package queue;

public class QueueX {
    private int []queueArray;
    private int maxsize;
    private int front;
    private int rear;
    private int nItems;

    public QueueX(int s) {
        this.maxsize = s;
        queueArray = new int[maxsize];
        this.front = 0;
        this.rear = -1;
        this.nItems = 0;
    }

    public void insert(int j) {
      if(rear == maxsize-1) {
          System.out.println("Queue is full");
      }
      else {
          queueArray[++rear] = j;
          nItems++;
      }
    }

    public int remove() {
        if(nItems == 0) {
            System.out.println("Queue is empty");
            return -1;
        } else {
            int temp = queueArray[front++];
            nItems--;
            return temp;
        }
    }

    public int peekFront() {
        if(nItems == 0) {
            System.out.println("Queue is empty");
            return -1;
        } else {
            return queueArray[front];
        }
    }


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

}
```

```java
import java. util.Scanner;

public class Main {
    public static void main(String[] args) {
        QueueX mainQueue = new QueueX(5);

        Scanner sc = new Scanner(System.in);

        for (int i=1; i<=5; i++) { //get keyboard inputs
            System.out.print("Enter transaction id : "+i+" ");
            int id=sc.nextInt();
            mainQueue.insert(id);
        }

        QueueX evenQueue = new QueueX(5);
        QueueX oddQueue = new QueueX(5);

        for (int i = 0;i<5 ;i++ ) {
            int value = mainQueue.remove();
            if(value % 2 ==0) {
                evenQueue.insert(value);
            }
            else {
                oddQueue.insert(value);
            }
        }

        System.out.println("Pc 2");
        while(!oddQueue.isEmpty()) {
            System.out.println("transaction " + oddQueue.remove());
        }

        System.out.println("Pc 1");
        while(!evenQueue.isEmpty()) {
            System.out.println("transaction " + evenQueue.remove());
        }
    }
}
```

02).
Characters given in a circular queue are stored in descending order: Write a java program to duplicate the same characters in ascending order and append to the same queue.
Simulate the above scenario by first entering 5 characters from the keyboard in ascending order. Store them in the queue.
Finally remove all the values from the queue and display them.

Before :

Z Y X W V

After :

Z Y X W V V W X Y Z

```java
package Stack;
import java.util.Scanner;

class Stack {
    private int maxSize;
    private int top;
    private char arrayStack[];

    public Stack(int size) {
        this.maxSize= size;
        this.top = -1;
        this.arrayStack = new char[maxSize];
    }

    public void push(char item) {
        if(isFull()) {
            System.out.println("Stack is full");
        } else {
            arrayStack[++top] = item;
        }
    }


    public char pop() {
        if(top == -1) {
            System.out.println("stack is empty");
            return '\0'; // more clear it's a null character
        }else {
            return arrayStack[top--];
        }
    }

    public boolean isEmpty() {
        return (top == -1);
    }
```

```java
    public boolean isFull() {
        return (maxSize - 1 == top);
    }

}
```

```
42 class Queue{
43     private int maxSize;
44     private int front;
45     private int rear;
46     private int noOfItems;
47     private char arrayQueue[];
48 
49     public Queue(int size) {
50         this.maxSize = size;
51         this.front = 0;
52         this.rear = -1;
53         this.noOfItems = 0;
54         this.arrayQueue = new char[maxSize];
55     }
56 
57     public void insert(char item) {
58         if(maxSize == noOfItems) {
59             System.out.println("queue is full");
60         }else {
61             if(rear == maxSize - 1) {
62                 rear = -1;
63             }
64             arrayQueue[++rear] = item;
65             noOfItems++;
66         }
67     }
68 
69     public char remove() {
70         if(noOfItems == 0) {
71             System.out.println("queue is empty");
72             return 0;
73         }else {
74             char temp = arrayQueue[front++];
75             if(front == maxSize){
76                 front = 0;
77             }
```

```
                noOfItems --;
                return temp;
            }
        }
    }
```

```java
class Main{
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);

        Queue q1  = new Queue(10);
        Stack s1 = new Stack (5);


        for(int i= 0 ; i<=4; i++) {
            System.out.print("Enter the character : ");
            char item = sc.next().charAt(0);
            q1.insert(item);
        }

        for(int i=0; i<=4; i++) {
            s1.push(q1.remove());
        }

        for(int i=0;i<=4 ; i++) {
            q1.insert(s1.pop());
        }

        Queue q2 = new Queue(5);

        for (int i=0; i<=4; i++) {
            char item = q1.remove();
            q2.insert(item);
            s1.push(item);
        }

        for (int i = 0; i<=4; i++) {
            q1.insert(q2.remove());
        }

        for (int i = 0; i<=4; i++) {
            q1.insert(s1.pop());
        }

        System.out.println("Final queue items : ");

        for (int i = 0; i<=9; i++) {
            System.out.println("Item : "+q1.remove());
        }


    }
}
```

- **Question**

a) Implement a **Node** class with suitable attributes to store employee number and name of employees.

b) Implement `displayNode ( )` method to display the details stored in a Node.

c) Implement the **Tree** class with the following data members and methods.

Tree
`Node root`

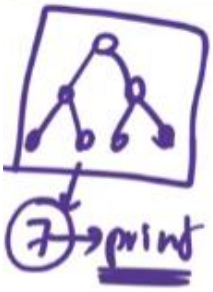
`Node find(int emp )`
`void insert(in emp, String name )`
`void inOrder( )`
`void preOrder( )`
`void postOrder( )`
`Node findRecursive( )`
`void deleteAll( )`

d) Implement a new method called `findRecursive( int emp)` which perform the find operation recursively.

e) Implement a method called `deleteAll ( )` to remove all the Nodes from the tree.

f) Write a application to do the following.

i) Create a tree of 10 Nodes with the following details.



Employee Number	Name
149	Anusha
167	Kosala
047	Dinusha
066	Mihiri
159	Jayani
118	Nimal
195	Nishantha
034	Avodya
105	Bimali
133	Sampath

ii) Display the employee data using inorder, preorder and postorder traversing.

iii) Allow the user to input any employee number from the keyboard and display the employee details if the employee exists in the tree.

iv) Delete all the nodes from the binary search tree.

v) Display the tree after deleting nodes.

```java
package Tree;
import java.util.Scanner;

class Node{
    public int empNo;
    public String empName;
    public Node leftChild;
    public Node rightChild;

    public Node (int empNo, String empName) {
        this.empNo = empNo;
        this.empName = empName;
        this.leftChild = null;
        this.rightChild = null;
    }

    public void displayNode() {
        System.out.println("Emp No : "+this.empNo+ "Emp name: "+this.empName);
    }
}

class Tree{
    private Node root;

    public Tree() {
        this.root = null;
    }

    public void insert(int empNo, String empName) {
        Node newNode = new Node(empNo, empName);

        if(root == null)
            root = newNode;
        else {
            Node current = root;
            Node parent;
            while (true) {
                parent = current;

                if(empNo < current.empNo) {
                    current = current.leftChild;
                    if(current == null) {
                        parent.leftChild = newNode;
                        return;
                    }
                }
                else {
                    current = current.rightChild;
                    if(current == null) {
                        parent.rightChild = newNode;
                        return;
                    }
                }
            }
        }
    }
```

```
60    public Node find(int key) {
61        Node current = root;
62        while (current.empNo != key)
63        {
64            if(key <current.empNo) {
65                current = current.leftChild;
66            }
67            else
68                current = current.rightChild;
69            if(current == null)
70                return null;
71        }
72        return current;
73    }
74
75    public void inOrder(Node localRoot) {
76        if(localRoot != null) {
77        inOrder(localRoot.leftChild);
78        localRoot.displayNode();
79        inOrder(localRoot.rightChild);
80        }
81    }
82
83    public void preOrder(Node localRoot) {
84        if(localRoot != null) {
85        localRoot.displayNode();
86        preOrder(localRoot.leftChild);
87        preOrder(localRoot.rightChild);
88        }
89    }
90
91    public void postOrder(Node localRoot) {
92        if(localRoot != null) {
93        postOrder(localRoot.leftChild);
94        postOrder(localRoot.rightChild);
95        localRoot.displayNode();
96        }
```

```
99    public Node getRoot() {
00        return root;
01    }
02
03    public Node findRecursive(int empNo) {
04        return findRecursiveHetlper(root, empNo);
05    }
06
07    private Node findRecursiveHetlper(Node localRoot, int empNo) {
08        if(localRoot == null) {
09        return null;
10        }
11        if(localRoot.empNo == empNo) {
12            return localRoot;
13        }
14        if(empNo <localRoot.empNo) {
15            return findRecursiveHetlper(localRoot.leftChild,empNo);
16        }else {
17            return findRecursiveHetlper(localRoot.rightChild,empNo);
18        }
19    }
20
21    public void deleteAll() {
22        root = null;
23    }
24 }
```

```java
public class TreeMain {
    public static void main(String Args[]) {
        Tree tree1 = new Tree();

        Scanner obj1 = new Scanner (System.in);

        tree1.insert(149,"Anusha");
        tree1.insert(167,"kosala");
        tree1.insert(047,"Dinusha");
        tree1.insert(066,"Mihiri");
        tree1.insert(134,"Jayani");
        tree1.insert(142,"Nidhantha");

        System.out.println("Inorder traversing: ");
        tree1.inOrder(tree1.getRoot());

        System.out.println("PreeOder traversing: ");
        tree1.preOrder(tree1.getRoot());

        System.out.println("Post order traversing: ");
        tree1.postOrder(tree1.getRoot());

        System.out.println("Employee number: ");
        int employeeNo = obj1.nextInt();

        Node item = tree1.findRecursive(employeeNo);

        if(item.empNo == employeeNo) {
            System.out.println("Name: "+item.empName+ "No: "+ item.empNo);
        }else {
            System.out.println("Item is not found ");
        }

        tree1.deleteAll();

        tree1.inOrder(tree1.getRoot());
    }
}
```

Characters given in a queue are stored in ascending order. Write a java program to store the characters in the **same queue** in descending order.

You need to first input the characters from the keyboard in ascending order and store in the queue.

After the values are stored in descending order, display the values of the queue by removing them.

Ex: Before

A	D	G	P	T

After

T	P	G	D	A

A stack class and a queue class are available in courseweb (Resources - version1F) . You can change the classes according to the requirement if needed.

Include your student ID number as a comment in the program

Activate Windows
Go to Settings to activate W

```
package test1;

import java.util.Scanner;


public class Main {

	public static void main(String[] args) {

		int arrSize = 5;

		Queue queueArr = new Queue(arrSize);
		Stack stackArr = new Stack(arrSize);

		Scanner sc = new Scanner(System.in);

		for (int i = 0; i < arrSize; i++) {

			System.out.print("Enter charater : ");
			char c = sc.next().charAt(0);
			queueArr.insert(c);
		}

		for (int i = 0; i < arrSize; i++) {
			System.out.println(queueArr.remove());
			stackArr.push(queueArr.remove());

		}
		for (int i = 0; i < arrSize; i++) {
			System.out.println(stackArr.pop());
			queueArr.insert(stackArr.pop());

		}
		for (int i = 0; i < arrSize; i++) {

			System.out.println(queueArr.remove());
		}
	}
}
```

A stack class and a queue class are available in courseweb (Resources - version1F) . You can change the classes according to the requirement if needed.

Include your student ID number as a comment in the program

Upload the program (*.java files) to the courseweb link "DSA_1F_ _<group>"

Grading Sheet:

Execution:

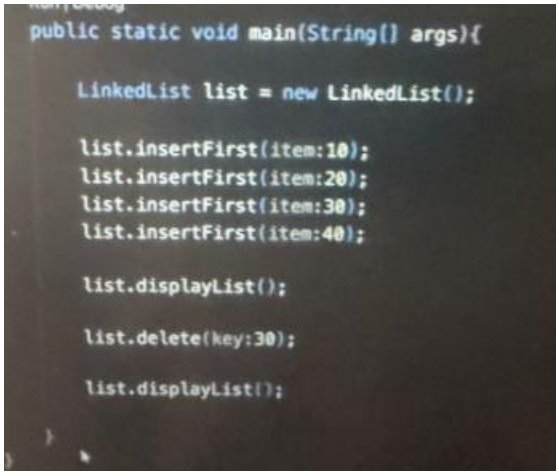
1) Program is compiling. **2 marks**
2) Program is running with correct results **3 marks**

Code:

3) Characters are stored in the queue **2 marks**
4) Characters are stored in descending order using proper data structures **10 mark**
5) Display the result from the queue - **3 marks**

Activate Windows

```
public class Main {

	public static void main(String[] args) {

		Queue queueX = new Queue(5);
		Stack stk = new Stack(5);

		Scanner sc = new Scanner(System.in);

		char num;

		for (int i = 1; i < 6; i++) {
			System.out.print("Enter number : ");
			char c = sc.next().charAt(0);
			queueX.insert(c);
		}

		while (!queueX.isEmpty()) {
			char value = queueX.remove();
			if (stk.isEmpty()) {
				stk.push(value);
			} else {
				while (stk.isEmpty() && value > stk.peek()) {
					queueX.insert(stk.pop());
				}

				stk.push(value);
			}
		}

		while (!stk.isEmpty()) {
			queueX.insert(stk.pop());
		}

		System.out.print("Integers in descending order: ");

		while (!queueX.isEmpty()) {
			System.out.print(queueX.remove() + " ");
		}
	}
```



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

    LinkedList list = new LinkedList();

    list.insertFirst(item:10);
    list.insertFirst(item:20);
    list.insertFirst(item:30);
    list.insertFirst(item:40);

    list.displayList();

    list.delete(key:30);

    list.displayList();

}
```