LAB SHEET -2

AIM 1: Understanding the concepts of Stack, Queue (10 points)

1. Write Java programs to implement a queue using an array.

```
1  package DSA;
2
3  public class Queue {
4
5     private int front, rear, capacity;
6     private int queue[];
7     public Queue(int size) {
8     front = rear = 0;
9     capacity = size;
10    queue = new int[capacity];
11    }
12    public void enqueue(int item) {
13    if(rear == capacity)
14    System.out.println("Queue is full!");
15    else {
16    queue[rear] = item;
17    rear++;
18    }
19    }
20    public void dequeue() {
21    if(front == rear) {
22    System.out.println("Queue is empty!");
```

Problems @ Javadoc Declaration Console

<terminated> Queue [Java Application] C:\Program Files\Java\jdk-1

```
Queue is empty!
Queue after enqueue
9 18 27 36
Queue is full!
Queue after dequeue
27 36
```

```
22      System.out.println("Queue is empty!");
23      }
24      else { for(int i=0;i<rear-1;i++) { queue[i] = queue[i+1];
25      }
26      rear--;
27      }
28      }
29      public void display() {
30      if(front == rear) {
31      System.out.println("Queue is empty!");
32      }
33      else
34      {
35      for(int i=front;i<rear;i++) {
36      System.out.print(queue[i]+" ");
37      }
38      }
39      }
40      public static void main(String[] args) {
41        Queue queue = new Queue(4);
42      System.out.println("Initial queue");
43      queue.display();
44      queue.enqueue(9);
45      queue.enqueue(18);
46      queue.enqueue(27);
47      queue.enqueue(36);
48      System.out.println("Queue after enqueue");
```

```
31    System.out.println("Queue is empty!");
32    }
33    else
34    {
35    for(int i=front;i<rear;i++) {
36    System.out.print(queue[i]+" ");
37    }
38    }
39    }
40    public static void main(String[] args) {
41      Queue queue = new Queue(4);
42    System.out.println("Initial queue");
43    queue.display();
44    queue.enqueue(9);
45    queue.enqueue(18);
46    queue.enqueue(27);
47    queue.enqueue(36);
48    System.out.println("Queue after enqueue");
49    queue.display();
50    System.out.println();
51    queue.enqueue(20);
52    queue.dequeue();
53    queue.dequeue();
54    System.out.println("Queue after dequeue");
55    queue.display();
56    }
57    }
```

2. Write a java program that reads an infix expression, converts the expression to postfix form and then evaluates the postfix expression (use stack ADT).

```java
package DSA;
   import java.util.ArrayDeque;
   import java.util.Deque;
   import java.util.Stack;
   class InfixToPostfix {
       // A utility function to return
      // precedence of a given operator
      // Higher returned value means
      // higher precedence
      static int Prec(char ch)
      {
         switch (ch) {
         case '+':
         case '-':
            return 1;

         case '*':
         case '/':
            return 2;

         case '^':
            return 3;
         }
         return -1;
      }

```

Problems @ Javadoc Declaration Console

<terminated> InfixToPostfix [Java Application] C:\Program Files\Java\

```
Theinfix to postfix expression will be 78563*3/+
```

```
25      }
26
27      //main method converts infix to postfix!!
28      static String infixToPostfix(String exp)
29      {
30
31          String result = new String("");
32
33          //empty stack
34          Deque<Character> stack
35              = new ArrayDeque<Character>();
36
37          for (int i = 0; i < exp.length(); ++i) {
38              char c = exp.charAt(i);
39
40
41              if (Character.isLetterOrDigit(c))
42                  result += c;
43              else if (c == '(')
44                  stack.push(c);
45              else if (c == ')') {
46                  while (!stack.isEmpty()
47                         && stack.peek() != '(') {
48                      result += stack.peek();
49                      stack.pop();
50                  }
```

Problems @ Javadoc Declaration Console ×

erminated> InfixToPostfix [Java Application] C:\Program Files\Java\jdk

```
ıeinfix to postfix expression will be 78563*3/+
```

```
52              }
53              else
54              {
55                  while (!stack.isEmpty()
56                          && Prec(c) <= Prec(stack.peek())) {
57
58                      result += stack.peek();
59                      stack.pop();
60                  }
61                  stack.push(c);
62              }
63          }
64          // pop all the operators from the stack
65          while (!stack.isEmpty()) {
66              if (stack.peek() == '(')
67                  return "Invalid Expression";
68              result += stack.peek();
69              stack.pop();
70          }
71          return result;
72      }
73      // Driver's code
74      public static void main(String[] args)
75      {
76          String exp = "The infix to postfix expression will be 78+(56*3)/3";
77            // Function call
78          System.out.println(infixToPostfix(exp));
```

```
74      public static void main(String[] args)
75      {
76          String exp = "The infix to postfix expression will be 78+(56*3)/3";
77            // Function call
78          System.out.println(infixToPostfix(exp));
79      }
80  }
```