

Subject: Algorithm and Data Structure Assignment 3

1. Implement a Stack using an array.

- **Test Case 1:**
Input: Push 5, 3, 7, Pop
Output: Stack = [5, 3], Popped element = 7
- **Test Case 2:**
Input: Push 10, Push 20, Pop, Push 15
Output: Stack = [10, 15], Popped element = 20

```
class Q1{
    int max = 10;
    int top;
    int arr[] = new int[max];

    Q1(){
        top = -1;
    }

    boolean isEmpty(){
        return (top < 0);
    }

    boolean push(int x){
        if(top > (max-1)){
            return false;
        }

        arr[++top] = x;
        return true;
    }

    void pop(){
        if(top < 0){
            System.out.println("Stack is Empty...");
            return;
        }
        else{
            System.out.println("Pop Element : " + arr[top--]);
        }
    }
}
```

```

void show(int index){
    if(index < 0){
        return;
    }
    else{
        System.out.println(arr[index] + " ");
        show (index - 1);
    }
}

void display(){
    if(isEmpty()){
        System.out.println("Stack is Empty...");
    }
    else{
        System.out.println("Stack Elements : ");
        show(top);
        System.out.println();
    }
}

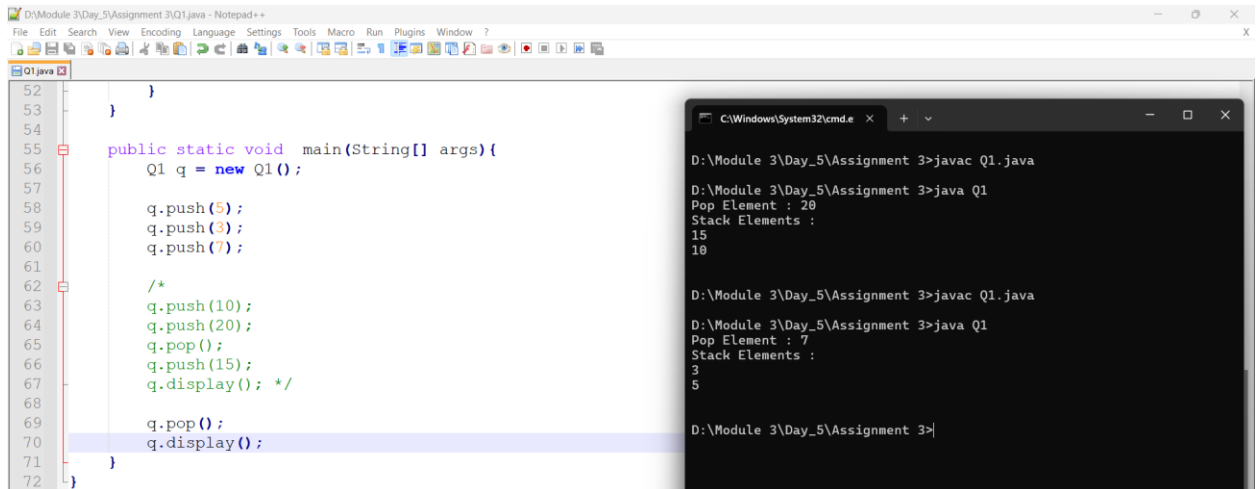
public static void main(String[] args){
    Q1 q = new Q1();

    q.push(5);
    q.push(3);
    q.push(7);

    /*
    q.push(10);
    q.push(20);
    q.pop();
    q.push(15);
    q.display(); */

    q.pop();
    q.display();
}
}

```



```
Q1.java
52     }
53 }
54
55 public static void main(String[] args){
56     Q1 q = new Q1();
57
58     q.push(5);
59     q.push(3);
60     q.push(7);
61
62     /*
63     q.push(10);
64     q.push(20);
65     q.pop();
66     q.push(15);
67     q.display(); */
68
69     q.pop();
70     q.display();
71 }
72 }
```

```
C:\Windows\System32\cmd.exe
D:\Module 3\Day_5\Assignment 3>javac Q1.java
D:\Module 3\Day_5\Assignment 3>java Q1
Pop Element : 20
Stack Elements :
15
10

D:\Module 3\Day_5\Assignment 3>javac Q1.java
D:\Module 3\Day_5\Assignment 3>java Q1
Pop Element : 7
Stack Elements :
3
5

D:\Module 3\Day_5\Assignment 3>
```

2. Check for balanced parentheses using a stack.

- **Test Case 1:**
Input: "{([()])}"
Output: Balanced
- **Test Case 2:**
Input: "([)]"
Output: Not Balanced

import java.util.*;

class BalancedParentheses{

```
@SuppressWarnings({ "rawtypes", "unchecked" })
public static boolean balanced(String input){
    Stack s = new Stack();

    char[] c = input.toCharArray();

    for(int i=0; i<c.length; i++)
    {
        char current = c[i];

        if (current == '(' || current == '[' || current == '{'){
            s.push(current);
            continue;
        }
        if(s.isEmpty()){
            return false;
        }

        char popchar;
        switch(current){
            case ')':
```

```

        popchar = (char)s.pop();
        if(popchar != '(')
            return false;
        break;
    case ']':
        popchar = (char)s.pop();
        if(popchar != '[')
            return false;
        break;
    case '}':
        popchar = (char)s.pop();
        if(popchar != '{')
            return false;
        break;
    }
}
return (s.isEmpty());
}

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

    System.out.print("Input : ");

    String s = sc.nextLine();

    System.out.print("Output : ");
    if(balanced(s)){
        System.out.println("Balanced");
    }
    else{
        System.out.println("Not Balanced");
    }
}
}

```

The screenshot shows a Java IDE with the file `BalancedParentheses.java` open. The code is as follows:

```

40     }
41     }
42     return (s.isEmpty());
43 }
44
45 public static void main(String[] args){
46     Scanner sc = new Scanner(System.in);
47
48     System.out.print("Input : ");
49
50     String s = sc.nextLine();
51
52     System.out.print("Output : ");
53     if(balanced(s)){
54         System.out.println("Balanced");
55     }
56     else{
57         System.out.println("Not Balanced");
58     }
59 }
60 }

```

To the right, a command prompt window shows the execution of the program:

```

D:\Module 3\Practise>javac BalancedParentheses.java
D:\Module 3\Practise>java BalancedParentheses
Input : ({[()]})
Output : Balanced
D:\Module 3\Practise>java BalancedParentheses
Input : ([)]
Output : Not Balanced
D:\Module 3\Practise>

```

3. Reverse a string using a stack.

- **Test Case 1:**

Input: "hello"

Output: "olleh"

- **Test Case 2:**

Input: "world"

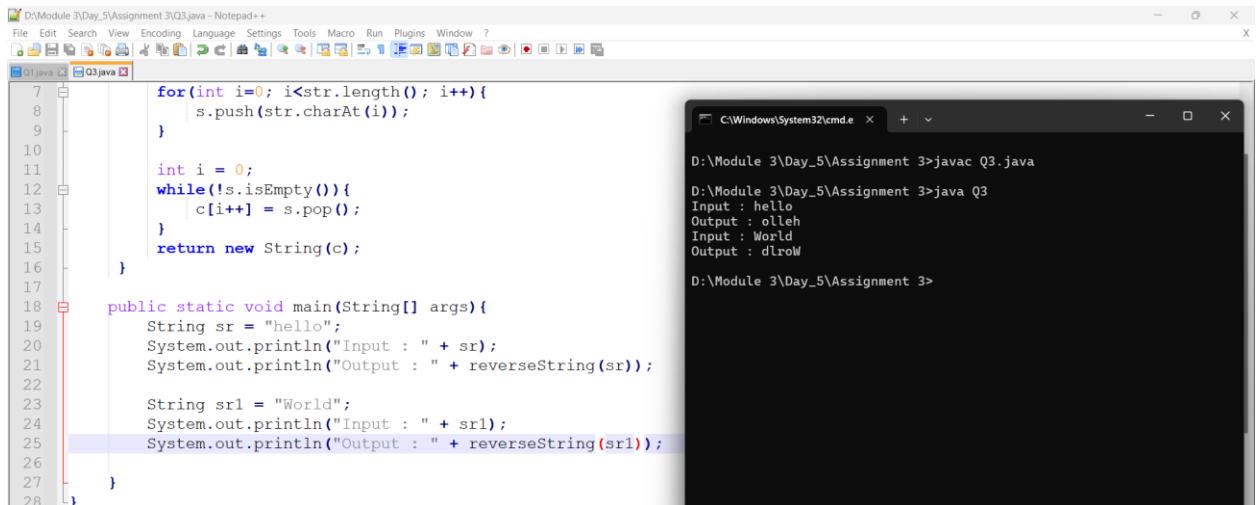
Output: "dlrow"

```
import java.io.*;
import java.util.*;
class Q3{
    public static String reverseString(String str){
        char[] c = new char[str.length()];
        Stack<Character> s = new Stack<Character>();
        for(int i=0; i<str.length(); i++){
            s.push(str.charAt(i));
        }

        int i = 0;
        while(!s.isEmpty()){
            c[i++] = s.pop();
        }
        return new String(c);
    }

    public static void main(String[] args){
        String sr = "hello";
        System.out.println("Input : " + sr);
        System.out.println("Output : " + reverseString(sr));

        String sr1 = "World";
        System.out.println("Input : " + sr1);
        System.out.println("Output : " + reverseString(sr1));
    }
}
```



```
7   for(int i=0; i<str.length(); i++){
8       s.push(str.charAt(i));
9   }
10
11   int i = 0;
12   while(!s.isEmpty()){
13       c[i++] = s.pop();
14   }
15   return new String(c);
16   }
17
18   public static void main(String[] args){
19       String sr = "hello";
20       System.out.println("Input : " + sr);
21       System.out.println("Output : " + reverseString(sr));
22
23       String srl = "World";
24       System.out.println("Input : " + srl);
25       System.out.println("Output : " + reverseString(srl));
26   }
27   }
28   }
```

```
D:\Module 3\Day_5\Assignment 3>javac Q3.java
D:\Module 3\Day_5\Assignment 3>java Q3
Input : hello
Output : olleh
Input : World
Output : dlroW
D:\Module 3\Day_5\Assignment 3>
```

4. Evaluate a postfix expression using a stack.

- **Test Case 1:**
Input: "5 3 + 2 *"
Output: 16
- **Test Case 2:**
Input: "4 5 * 6 /"
Output: 3

```
import java.util.*;
```

```
class Q4{
    @SuppressWarnings({"rawtypes", "unchecked"})
    public static int postexp(String str){
        Stack<Integer> st = new Stack<>();

        for(String s : str.split(" ")){
            if(isNumeric(s)){
                st.push(Integer.parseInt(s));
            }
            else{
                int op1 = st.pop();
                int op2 = st.pop();

                switch(s){
                    case "+":
                        st.push(op2 + op1);
                        break;
                    case "-":
                        st.push(op2 - op1);
                        break;
                    case "*":
                        st.push(op2 * op1);
```

```

                break;
            case "/":
                st.push(op2 / op1);
                break;
        }
    }
}
return st.pop();
}

private static boolean isNumeric(String str){
    return str.matches("-?\\d+(\\.\\d+)?");
}

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

    System.out.print("Input : ");
    String s = sc.nextLine();

    System.out.println("Output : " + postexp(s));
}
}

```

The screenshot shows a Notepad++ window with a Java file named Q4.java. The code is a Java program that evaluates postfix expressions using a stack. It defines a method `postexp` (partially visible) and a `main` method that prompts the user for input and prints the output. The Command Prompt window shows the execution of the program:

```

D:\Module 3\Day_5\Assignment 3>javac Q4.java
D:\Module 3\Day_5\Assignment 3>java Q4
Input : 4 5 * 6 /
Output : 3
D:\Module 3\Day_5\Assignment 3>java Q4
Input : 5 3 + 2 *
Output : 16
D:\Module 3\Day_5\Assignment 3>

```

5. Convert an infix expression to postfix using a stack.

- **Test Case 1:**
Input: "A + B * C"
Output: "A B C * +"
- **Test Case 2:**
Input: "A * B + C / D"
Output: "A B * C D / +"

① Input: A + B * C
 - A + B * C
 - A B + * C
 - A B + C *
 - A B C * +
 output: A B C * +

② i/p: A * B + C / D
 A B * + C / D
 A B * + C D /
 A B * C D / +
 o/p: - A B * C D / +

6. Implement a Queue using an array.

- Test Case 1:

Input: Enqueue 5, Enqueue 10, Dequeue

Output: Queue = [10], Dequeued element = 5

- Test Case 2:

Input: Enqueue 1, 2, 3, Dequeue, Dequeue

Output: Queue = [3], Dequeued elements = 1, 2

```
class Q6{
    int size = 5;
    int Q[] = new int[size];
    int front, rear;

    public Q6(){
        front = -1;
        rear = -1;
    }

    boolean isEmpty(){
        return (front == -1 || front > rear);
    }

    boolean isFull(){
        return (rear == size-1);
    }

    void enqueue(int x){
```



```

        if(isFull()){
            System.out.println("Queue is Full.");
        }
        else {
            if(front == -1){
                front = 0;
            }
            rear++;
            Q[rear] = x;
            System.out.println(x);
        }
    }

    void dequeue(){
        if(isEmpty()){
            System.out.println("Queue is Empty..");
        }
        else{
            System.out.println(Q[front] + " removed.");
            front++;
            if(front > rear){
                front = -1;
                rear = -1;
            }
        }
    }

    void display(){
        if(isEmpty()){
            System.out.println("Queue is Empty");
        }
        else{
            System.out.println("Queue elements : ");
            for(int i=front; i<=rear; i++){
                System.out.print(Q[i] + " ");
                if (i < rear) {
                    System.out.print(", ");
                }
            }
            System.out.println();
        }
    }

    public static void main(String[] args){
        Q6 q = new Q6();
        Q6 q1 = new Q6();
    }

```



```

int front, rear;

Q7(){
    front = -1;
    rear = -1;
}

boolean isEmpty(){
    return (front == -1);
}

boolean isFull(){
    return ((rear + 1) % size == front);
}

void enqueue(int x){
    if(isFull()){
        System.out.println("Queue is full.");
    }
    else {
        if(front == -1){
            front = 0;
        }
        Q[++rear] = x;
        System.out.println(x);
    }
}

void dequeue(){
    if(isEmpty()){
        System.out.println("Queue is empty.");
    }
    else{
        System.out.println("The element " + Q[front] + " is removed");
        front++;
        if(front > rear){
            front = -1;
            rear = -1;
        }
    }
}

void display(){
    if(isEmpty()){
        System.out.println("Queue is empty...");
    }
}

```

```

        else{
            System.out.println("Queue Element : ");
            for(int i=front; i<=rear; i++){
                System.out.print(Q[i] + " ");
            }
            System.out.println();
        }
    }

    public static void main(String[] args){
        Q7 q = new Q7();
        q.enqueue(4);
        q.enqueue(5);
        q.enqueue(6);
        q.enqueue(7);
        q.dequeue();
        q.enqueue(8);

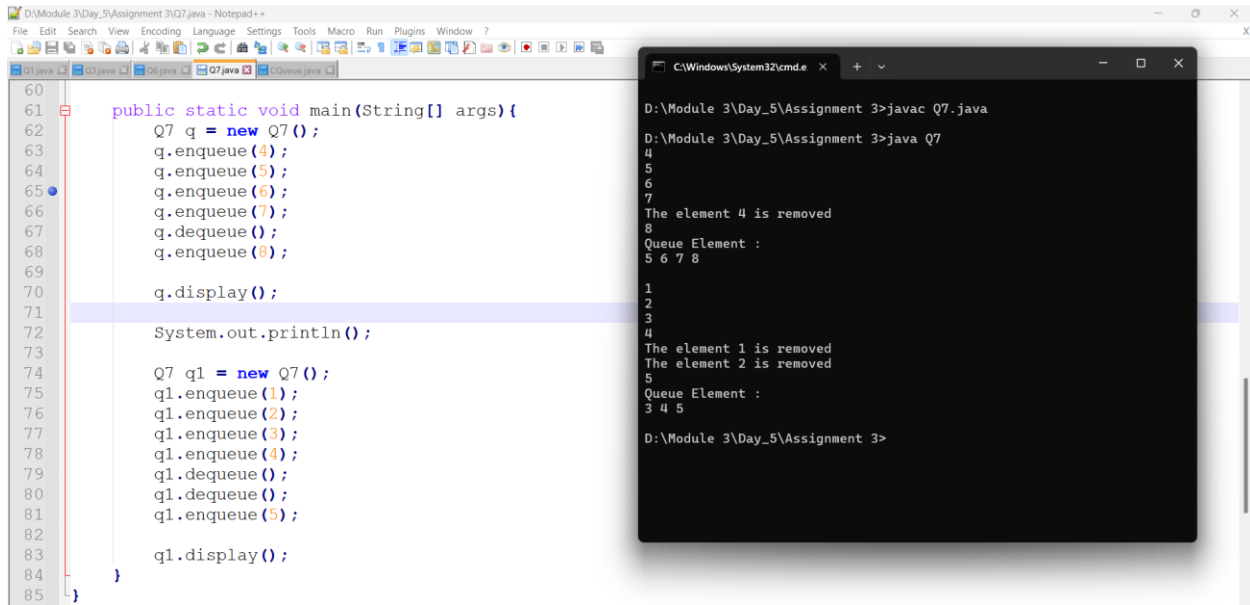
        q.display();

        System.out.println();

        Q7 q1 = new Q7();
        q1.enqueue(1);
        q1.enqueue(2);
        q1.enqueue(3);
        q1.enqueue(4);
        q1.dequeue();
        q1.dequeue();
        q1.enqueue(5);

        q1.display();
    }
}

```



```
60
61 public static void main(String[] args){
62     Q7 q = new Q7();
63     q.enqueue(4);
64     q.enqueue(5);
65     q.enqueue(6);
66     q.enqueue(7);
67     q.dequeue();
68     q.enqueue(8);
69
70     q.display();
71
72     System.out.println();
73
74     Q7 q1 = new Q7();
75     q1.enqueue(1);
76     q1.enqueue(2);
77     q1.enqueue(3);
78     q1.enqueue(4);
79     q1.dequeue();
80     q1.dequeue();
81     q1.enqueue(5);
82
83     q1.display();
84
85 }
```

```
D:\Module 3\Day_5\Assignment 3>javac Q7.java
D:\Module 3\Day_5\Assignment 3>java Q7
4
5
6
7
The element 4 is removed
8
Queue Element :
5 6 7 8

1
2
3
4
The element 1 is removed
The element 2 is removed
5
Queue Element :
3 4 5

D:\Module 3\Day_5\Assignment 3>
```

8. Implement a Queue using two Stacks.

- **Test Case 1:**
Input: Enqueue 3, Enqueue 7, Dequeue
Output: Queue = [7], Dequeued element = 3
- **Test Case 2:**
Input: Enqueue 10, 20, Dequeue, Dequeue
Output: Queue = [], Dequeued elements = 10, 20

```
import java.util.*;
```

```
class Q8{
```

```
    Stack<Integer> s1 = new Stack<>();
    Stack<Integer> s2 = new Stack<>();

    void enqueue(int data){
        s1.push(data);
    }

    int deque(){
        if(s2.isEmpty()){
            if(s1.isEmpty()){
                System.out.println("Empty");
                return -1;
            }
        }
        while(!s1.isEmpty()){
            s2.push(s1.pop());
        }
    }
}
```

```

        }
        return s2.pop();
    }

    void display(){
        if(!s2.isEmpty()){
            System.out.println("Queue : " + s2);
        }
        else{
            System.out.println("Queue : " + s1);
        }
    }

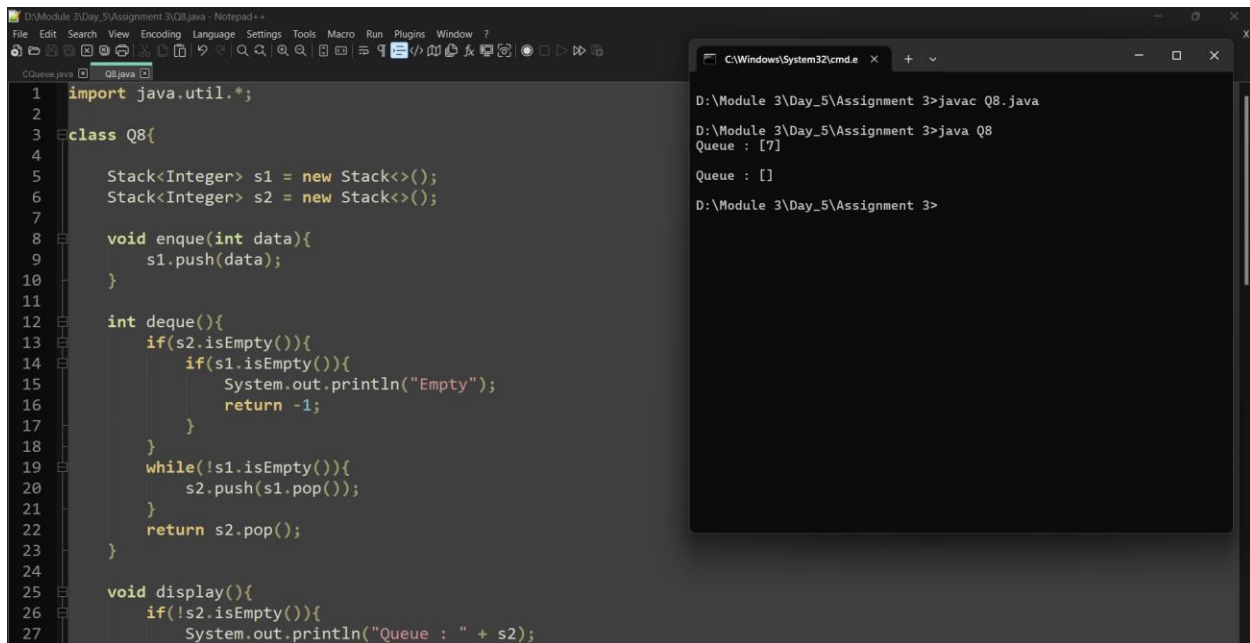
    public static void main(String[] args){
        Q8 q = new Q8();

        q.enqueue(3);
        q.enqueue(7);
        q.dequeue();
        q.display();

        System.out.println();

        Q8 q1 = new Q8();
        q1.enqueue(10);
        q1.enqueue(20);
        q1.dequeue();
        q1.dequeue();
        q1.display();
    }
}

```



```
1 import java.util.*;
2
3 class Q8{
4
5     Stack<Integer> s1 = new Stack<>();
6     Stack<Integer> s2 = new Stack<>();
7
8     void enqueue(int data){
9         s1.push(data);
10    }
11
12    int dequeue(){
13        if(s2.isEmpty()){
14            if(s1.isEmpty()){
15                System.out.println("Empty");
16                return -1;
17            }
18        }
19        while(!s1.isEmpty()){
20            s2.push(s1.pop());
21        }
22        return s2.pop();
23    }
24
25    void display(){
26        if(!s2.isEmpty()){
27            System.out.println("Queue : " + s2);
28        }
29    }
30 }
```

```
D:\Module 3\Day_5\Assignment 3>javac Q8.java
D:\Module 3\Day_5\Assignment 3>java Q8
Queue : []
D:\Module 3\Day_5\Assignment 3>
```

11. Sort an array using a heap (Heap Sort).

- **Test Case 1:**

Input: [5, 1, 12, 3, 9]

Output: [1, 3, 5, 9, 12]

- **Test Case 2:**

Input: [20, 15, 8, 10]

Output: [8, 10, 15, 20]

```
class Heapsort{
    void heapify(int arr[], int n, int i){
        int largest = i; // Root
        int l = 2*i + 1; // LC
        int r = 2*i + 2; // RC

        if(l<n && arr[l] > arr[largest])
            largest = l;
        if(r<n && arr[r] > arr[largest])
            largest = r;

        if(largest != i){
            int temp = arr[i];
            arr[i] = arr[largest];
            arr[largest] = temp;
            heapify(arr, n, largest);
        }
    }

    void heapsort(int arr[]){
        System.out.println();
    }
}
```

```

        int n = arr.length;
        for(int i=n/2-1; i>=0; i--){
            heapify(arr, n, i);
        }
        for(int i=n-1; i>0;i--){
            int temp = arr[0];
            arr[0] = arr[i];
            arr[i] = temp;
            heapify(arr, i, 0);
        }
    }

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

    public static void main(String[] args){
        Heapsort h = new Heapsort();
        int a[] = {5, 1, 12, 3, 9};
        System.out.print("Input : ");
        h.display(a);
        h.heapsort(a);
        System.out.print("Output : ");
        h.display(a);

        System.out.println();
        System.out.println();

        int ar[] = {20, 15, 8, 10};
        System.out.print("Input : ");
        h.display(ar);
        h.heapsort(ar);
        System.out.print("Output : ");
        h.display(ar);
    }
}

```


The image shows a Notepad++ window with a Java file named `Heapsort.java`. The code implements a Heapsort algorithm. It has a `main` method that takes an array of integers and sorts it in ascending order. The code is as follows:

```
43
44 public static void main(String[] args){
45     Heapsort h = new Heapsort();
46     int a[] = {5, 1, 12, 3, 9};
47     System.out.print("Input : ");
48     h.display(a);
49     h.heapsort(a);
50     System.out.print("Output : ");
51     h.display(a);
52
53     System.out.println();
54     System.out.println();
55
56     int ar[] = {20, 15, 8, 10};
57     System.out.print("Input : ");
58     h.display(ar);
59     h.heapsort(ar);
60     System.out.print("Output : ");
61     h.display(ar);
62 }
63 }
```

To the right of the Notepad++ window is a Windows command prompt window. It shows the command `javac Heapsort.java` being executed, followed by `java Heapsort`. The output of the program is displayed in the command prompt:

```
D:\Module 3\Day_4>javac Heapsort.java
D:\Module 3\Day_4>java Heapsort
Input :
5 1 12 3 9
Output :
1 3 5 9 12

Input :
20 15 8 10
Output :
8 10 15 20
D:\Module 3\Day_4>
```

15. Design a Circular Queue with a fixed size, supporting enqueue, dequeue, and isFull/isEmpty operations.

- **Test Case 1:**
Input: Size = 4, Enqueue 1, 2, 3, 4, isFull()
Output: True
- **Test Case 2:**
Input: Size = 3, Enqueue 5, 6, Dequeue, Enqueue 7, isEmpty()
Output: False

```
class Q15{
    int[] Q;
    int front, rear, size, capacity;

    public Q15(int capacity){
        this.capacity = capacity;
        Q = new int[capacity];
        front = -1;
        rear = -1;
        size = 0;
    }

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

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

    void enqueue(int x){
        if(isFull()){
            System.out.println("Queue is full");
        }
    }
}
```

```

    }
    else{
        if(front == -1){
            front = 0;
        }
        rear = (rear+1) % capacity;
        Q[rear] = x;
        size++;
        System.out.println(x);
    }
}

void dequeue(){
    if(isEmpty()){
        System.out.println("Queue is Empty");
        return;
    }
    else{
        System.out.println("The element " + Q[front] + " is removed.");
        front = (front+1) % capacity;
        size--;
    }
}

void display(){
    if(isEmpty()){
        System.out.println("Queue is Empty");
    }
    else {
        System.out.println("Queue Elements :");
        for(int i=front; i<=rear; i++){
            System.out.print(Q[i] + " --> ");
        }
        System.out.println();
    }
}

public static void main(String[] args){
    Q15 q = new Q15(4);
    q.enqueue(1);
    q.enqueue(2);
    q.enqueue(3);
    q.enqueue(4);

    System.out.println("Output : " + q.isFull());
    System.out.println();

    Q15 q1 = new Q15(3);

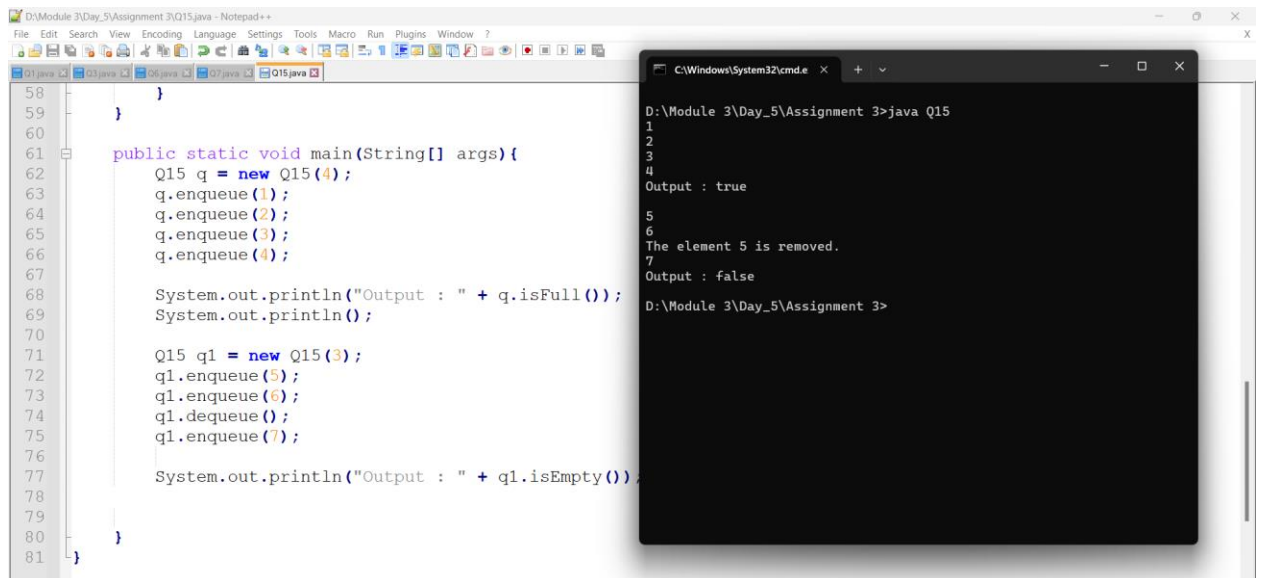
```

```
q1.enqueue(5);
q1.enqueue(6);
q1.dequeue();
q1.enqueue(7);
```

```
System.out.println("Output : " + q1.isEmpty());
```

```
}
```

```
}
```



The screenshot displays a Notepad++ window with a Java file named Q15.java. The code defines a queue class Q15 and a main method. The main method first creates a Q15 object 'q' with a capacity of 4, enqueues elements 1, 2, 3, and 4, and prints 'Output : ' + q.isFull()' followed by a newline. Then, it creates another Q15 object 'q1' with a capacity of 3, enqueues elements 5, 6, dequeues element 5, enqueues element 7, and prints 'Output : ' + q1.isEmpty()' followed by a newline. A Windows command prompt window is overlaid on the right, showing the execution of 'java Q15' from the directory 'D:\Module 3\Day_5\Assignment 3'. The output of the program is shown as follows:

```
D:\Module 3\Day_5\Assignment 3>java Q15
1
2
3
4
Output : true
5
6
The element 5 is removed.
7
Output : false
D:\Module 3\Day_5\Assignment 3>
```