Assignment Questions 15

Question 1.

Given an array arr[ ] of size N having elements, the task is to find the next greater element for each element of the array in order of their appearance in the array.Next greater element of an element in the array is the nearest element on the right which is greater than the current element.If there does not exist next greater of current element, then next greater element for current element is -1. For example, next greater of the last element is always -1.

Example 1:

Input:

N = 4, arr[] = [1 3 2 4]

Output:

3 4 4 -1

Explanation:

In the array, the next larger element
to 1 is 3 , 3 is 4 , 2 is 4 and for 4 ?
since it doesn’t exist, it is -1.

Example 2:

Input:

N = 5, arr[] [6 8 0 1 3]

Output:

8 -1 1 3 -1

Explanation:

In the array, the next larger element to
6 is 8, for 8 there is no larger elements
hence it is -1, for 0 it is 1 , for 1 it
is 3 and then for 3 there is no larger
element on right and hence -1.

code:-

```
class Main {
 static void printNGE(int arr[], int n)
 {
  int next, i, j;
  for (i = 0; i < n; i++) {
   next = -1;
   for (j = i + 1; j < n; j++) {
    if (arr[i] < arr[j]) {
     next = arr[j];
     break;
    }
   }
   System.out.println(arr[i] + " -- " + next);
  }
 }

 public static void main(String args[])
 {
  int arr[] = { 1 3 2 4 };
  int n = arr.length;
  printNGE(arr, n);
 }
}
```

Question 2

Given an array a of integers of length n, find the nearest smaller number for every element such that the s maller element is on left side.If no small element present on the left print -1.

Example 1:

Input: n = 3

a = {1, 6, 2}

Output: -1 1 1

Explaination: There is no number at the left of 1. Smaller number than 6 and 2 is 1.

Example 2:

Input: n = 6

a = {1, 5, 0, 3, 4, 5}

Output: -1 1 -1 0 3 4

Explaination: Upto 3 it is easy to see the smaller numbers. But for 4 the smaller numbers are 1, 0 and 3. But among them 3 is closest. Similary for 5 it is 4.

code:-

```
class GFG {
 static void printPrevSmaller(int[] arr, int n)
 {
  System.out.print("_, ");
  for (int i = 1; i < n; i++) {
  int j;
   for (j = i - 1; j >= 0; j--) {
    if (arr[j] < arr[i]) {
     System.out.print(arr[j] + ", ");
     break;
    }
   }
   if (j == -1)
    System.out.print("_, ");
  }
 }
 public static void main(String[] args)
 {
  int[] arr = { 1, 3, 0, 2, 5 };
  int n = arr.length;
  printPrevSmaller(arr, n);
 }
}
```

Question 3.

Implement a Stack using two queues q1 and q2.

Example 1:

Input:

push(2)

push(3)

pop()

push(4)

pop()

Output:3 4

Explanation:

push(2) the stack will be {2}
push(3) the stack will be {2 3}
pop() poped element will be 3 the
       stack will be {2}
push(4) the stack will be {2 4}
pop() poped element will be 4

Example 2:
Input:
push(2)
pop()
pop()
push(3)
Output:2 -1

code:-

```
class GfG {
 static class Stack {
  static Queue<Integer> q1= new LinkedList<Integer>();
  static Queue<Integer> q2= new LinkedList<Integer>();
  static int curr_size;

  static void push(int x)
  {
   q2.add(x);
   while (!q1.isEmpty()) {
    q2.add(q1.peek());
    q1.remove();
   }
   Queue<Integer> q = q1;
   q1 = q2;
   q2 = q;
  }

  static void pop()
  {
   if (q1.isEmpty())
   return;
   q1.remove();
  }

  static int top()
  {
   if (q1.isEmpty())
    return -1;
   return q1.peek();
  }

  static int size() { return q1.size(); }
 }
 public static void main(String[] args)
 {
  Stack s = new Stack();
  s.push(1);
```

```
  s.push(2);
  s.push(3);
  System.out.println("current size: " + s.size());
  System.out.println(s.top());
  s.pop();
  System.out.println(s.top());
  s.pop();
  System.out.println(s.top());
  System.out.println("current size: " + s.size());
 }
}
```

Question 4.

You are given a stack St. You have to reverse the stack using recursion.

Example 1:

Input:St = {3,2,1,7,6}

Output:{6,7,1,2,3}

Example 2:

Input:St = {4,3,9,6}

Output:{6,9,3,4}

code:-

```
class Test {
 static Stack<Character> st = new Stack<>();
 static void insert_at_bottom(char x)
 {

  if (st.isEmpty())
   st.push(x);

  else {
   char a = st.peek();
   st.pop();
   insert_at_bottom(x);
   st.push(a);
  }
 }
 static void reverse()
 {
  if (st.size() > 0) {
   char x = st.peek();
   st.pop();
   reverse();
   insert_at_bottom(x);
  }
 }
 public static void main(String[] args)
 {
  st.push('1');
  st.push('2');
  st.push('3');
  st.push('4');
  System.out.println("Original Stack");
```

```
  System.out.println(st);
  reverse();

  System.out.println("Reversed Stack");

  System.out.println(st);
 }
}
```

Question 5.
You are given a string S, the task is to reverse the string using stack.
Example 1:
Input: S="GeeksforGeeks"
Output: skeeGrofskeeG

code:-

```
class Stack {
 int size;
 int top;
 char[] a;
 boolean isEmpty() { return (top < 0); }

 Stack(int n)
 {
  top = -1;
  size = n;
  a = new char[size];
 }
 boolean push(char x)
 {
  if (top >= size) {
   System.out.println("Stack Overflow");
   return false;
  }
  else {
   a[++top] = x;
   return true;
  }
 }
 char pop()
 {
  if (top < 0) {
   System.out.println("Stack Underflow");
   return 0;
  }
  else {
   char x = a[top--];
   return x;
  }
 }
}

class Main {
```

```
 public static void reverse(StringBuffer str)
 {
  int n = str.length();
  Stack obj = new Stack(n);
  int i;
  for (i = 0; i < n; i++)
  obj.push(str.charAt(i));
  for (i = 0; i < n; i++) {
  char ch = obj.pop();
  str.setCharAt(i, ch);
  }
 }
 public static void main(String args[])
 {
  StringBuffer s = new StringBuffer("GeeksQuiz");
  reverse(s);
  System.out.println("Reversed string is " + s);
 }
}
```

Question 6.

Given string S representing a postfix expression, the task is to evaluate the expression and find the final v alue. Operators will only include the basic arithmetic operators like *, /, + and -.

Example 1:

Input: S = "231*+9-"

Output: -4

Explanation:

After solving the given expression,

we have -4 as result.

Example 2:

Input: S = "123+*8-"

Output: -3

Explanation:

After solving the given postfix

expression, we have -3 as result.

code:-

```
public class Test {
 static int evaluatePostfix(String exp)
 {
  Stack<Integer> stack = new Stack<>();
  for (int i = 0; i < exp.length(); i++) {
  char c = exp.charAt(i);
   if (Character.isDigit(c))
   stack.push(c - '0');
   else {
    int val1 = stack.pop();
    int val2 = stack.pop();

    switch (c) {
    case '+':
     stack.push(val2 + val1);
     break;
    case '-':
```

```
     stack.push(val2 - val1);
     break;
    case '/':
     stack.push(val2 / val1);
     break;
    case '*':
     stack.push(val2 * val1);
     break;
    }
   }
  }
  return stack.pop();
 }
 public static void main(String[] args)
 {
  String exp = "231*+9-";
  System.out.println("postfix evaluation: "+ evaluatePostfix(exp));
 }
}
```

Question 7.
Design a stack that supports push, pop, top, and retrieving the minimum element in constant time.
Implement the ‘MinStack‘ class:

- ‘MinStack()‘ initializes the stack object.
- ‘void push(int val)‘ pushes the element ‘val‘ onto the stack.
- ‘void pop()‘ removes the element on the top of the stack.
- ‘int top()‘ gets the top element of the stack.
- ‘int getMin()‘ retrieves the minimum element in the stack.

You must implement a solution with ‘O(1)‘ time complexity for each function.

Example 1:
Input
["MinStack","push","push","push","getMin","pop","top","getMin"]
[[],[-2],[0],[-3],[],[],[],[]]

Output
[null,null,null,null,-3,null,0,-2]

Explanation
MinStack minStack = new MinStack();
minStack.push(-2);
minStack.push(0);
minStack.push(-3);
minStack.getMin(); // return -3
minStack.pop();
minStack.top();   // return 0
minStack.getMin(); // return -2

code:-

```
class MinStack {
    LinkedList<TplusMin> stack;
    private class TplusMin {
```

```
        int val;
        int min;
        public TplusMin(int val, int min) {
            this.val = val;
            this.min = min;
        }
    }

    public MinStack() {
        stack = new LinkedList<>();
    }

    public void push(int val) {
        int newMin;
        if (stack.size() == 0){
            newMin = val;
        }
        else {
            int currentMin = stack.getFirst().min;
            newMin = val < currentMin ? val : currentMin;
        }
        stack.addFirst(new TplusMin(val, newMin));
    }

    public void pop() {
        stack.removeFirst();
    }

    public int top() {
        return stack.peekFirst().val;
    }

    public int getMin() {
        return stack.peekFirst().min;
    }
}
```

Question 8.
Given n non-negative integers representing an elevation map where the width of each bar is 1, compute h ow much water it can trap after raining.
Example 1:
Input: height = [0,1,0,2,1,0,1,3,2,1,2,1]
Output: 6
Explanation: The above elevation map (black section) is represented by array [0,1,0,2,1,0,1,3,2,1,2,1]. In t his case, 6 units of rain water (blue section) are being trapped.

Example 2:
Input: height = [4,2,0,3,2,5]
Output: 9

code:-

```
class GFG {
 public static int maxWater(int[] arr, int n)
 {
```

```
  int res = 0;
  for (int i = 1; i < n - 1; i++) {
   int left = arr[i];
   for (int j = 0; j < i; j++) {
    left = Math.max(left, arr[j]);
   }
   int right = arr[i];
   for (int j = i + 1; j < n; j++) {
    right = Math.max(right, arr[j]);
   }
   res += Math.min(left, right) - arr[i];
  }
  return res;
 }
 public static void main(String[] args)
 {
  int[] arr = { 0, 1, 0, 2, 1, 0, 1, 3, 2, 1, 2, 1 };
  int n = arr.length;
  System.out.print(maxWater(arr, n));
 }
}
```