Assignment on Queue and Stack

Jakkanagari Harika-20BD1A1290

Program 1:

IMPLEMENT A STACK USING TWO QUEUES Q1 AND Q2.

Example 1:

Input: push(2)

push(3)

pop()

push(4)

pop()

Output: 3 4

Solution:

import java.util.\*;

class SwthtwoQueues{  
 public static void main(String[] args){

Stack s=new Stack();

s.push(2);

s.push(3);

System.out.println(s.pop());

s.push(4);

System.out.println(s.pop());

}

Public static class Stack{

static Queue<Integer> q1=new Queue<Integer>();

static Queue<Integer> q2=new Queue<Integer>();

static int cs;

public static void push(int ele){

q2.add(ele);

while(!q1.isEmpty()){

q2.add(q1.peek());

q1.remove();

}

Queue<Integer> q=q1;

q1=q2;

q2=q;

}

Public static void pop(){  
 if(q1.isEmpty()){

Return;

}

Return q1.peek();

}

}

}

Program 2:

IMPLEMENT A QUEUE USING 2 STACKS S1 AND S2 .

A Query Q is of 2 Types (i) 1 x (a query of this type means pushing 'x' into the queue) (ii) 2 (a query of this type means to pop element from queue and print the poped element)

Solution:

Import java.util.\*;

class QwthtwoS{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int a[]=new int[n];

Queue q=new Queue();

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

int q=sc.nextInt(){

if(q==1){  
 int x=sc.nextInt();

q.enqueue(x);

}

else if(q==2){

System.out.println(q.dequeue());

}

}

}

Static class Queue{

static Stack<Integer> s1=new Stack<>();

static Stack<Integer> s2=new Stack<>();

public static void enqueue(int x){

while(!s1.isEmpty()){

s2.push(s1.pop());

}

s1.push(x);

While(!s2.isEmpty())[  
 s1.push(s2.pop());

}

}

public static int dequeue(){

if(!s1.isEmpty()){

return s1.pop();

}

Return -1;

}

}

Example :

Input:5 1 2 1 3 2 1 4 2

Output: 2 3

Program 3:

GIVEN AN ARRAY ARR[] OF SIZE N AND AN INTEGER K. FIND THE MAXIMUM FOR EACH AND EVERY CONTIGUOUS SUBARRAY OF SIZE K.

Solution:

import java.util.\*;

class Main{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

Stack<Integer> s=new Stack<>();

int n=sc.nextInt();

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

s.push(sc.nextInt());

}

int k=sc.nextInt();

List<Integer> ll=new ArrayList<>();

System.out.println(method(s,k,ll));

}

public static List<Integer> method(Stack<Integer> s,int k,List<Integer> ll){

if(s.size()<k){

return ll;

}

int sum=s.get(0);

for(int i=1;i<k;i++){

sum=Math.max(sum,s.get(i));

}

s.remove(0);

ll.add(sum);

return method(s,k,ll);

}

}

Example :

Input:

N = 9, K = 3

arr[] = 1 2 3 1 4 5 2 3 6

Output: 3 3 4 5 5 5 6

Program 6:

GIVEN A STRING CONTAINING ONLY PARENTHESES, DETERMINE IF THE STRING IS VALID. AN INPUT STRING IS VALID IF:

Solution:

import java.util.\*;

public class Main {

static boolean isValid(String s)

{

Deque<Character> stack= new ArrayDeque<Character>();

for (int i = 0; i < s.length(); i++) {

char x = s.charAt(i);

if (x == '(' || x == '[' || x == '{') {

stack.push(x);

continue;

}

if (stack.isEmpty())

return false;

char check;

switch (x) {

case ')':

check = stack.pop();

if (check == '{' || check == '[')

return false;

break;

case '}':

check = stack.pop();

if (check == '(' || check == '[')

return false;

break;

case ']':

check = stack.pop();

if (check == '(' || check == '{')

return false;

break;

}

}

return (stack.isEmpty());

}

public static void main(String[] args)

{

Scanner sc=new Scanner(System.in);

String s = sc.next();

if (isValid(s))

System.out.println("True");

else

System.out.println("False");

}

}

Example

input: "()[]{}"

Expected output: True

PROBLEM 7

. GIVEN AN ARRAY, FIND THE NEXT GREATER ELEMENT (NGE) FOR EVERY ELEMENT IN THE ARRAY. THE NEXT GREATER ELEMENT FOR AN ELEMENT X IS THE FIRST GREATER ELEMENT ON THE RIGHT SIDE OF X IN THE ARRAY. IF THERE IS NO GREATER ELEMENT ON THE RIGHT SIDE, THEN THE OUTPUT FOR THAT ELEMENT SHOULD BE -1.

Solution:  
import java.util.\*;

public class Main{

public static void main(String[] args) {

Map<Integer, Integer> map = new HashMap<>();

Stack<Integer> stack = new Stack<>();

for (int i = arr.length - 1; i >= 0; i--) {

while (!stack.isEmpty() && arr[i] > stack.peek()) {

stack.pop();

}

map.put(arr[i], stack.isEmpty() ? -1 : stack.peek());

stack.push(arr[i]);

}

for (int i = 0; i < arr.length; i++) {

System.out.println("Next greater element for " + arr[i] + ": " + map.get(arr[i]));

}

}

}

Program 8:

GIVEN A STRING S CONSISTING OF LOWERCASE LETTERS, REMOVE ADJACENT DUPLICATES FROM S. THE FINAL OUTPUT SHOULD BE IN LEXICOGRAPHICALLY SMALLEST ORDER. Example input: "abbaca" Expected output: "ca".

Solution

import java.util.\*;

public class Test {

public static void main(String[] args) {

String s = sc.nextInt();

Stack<Character> stack = new Stack<>();

for (int i = 0; i < s.length(); i++) {

char c = s.charAt(i);

if (!stack.isEmpty() && stack.peek() == c) {

stack.pop();

} else {

stack.push(c);

}

}

StringBuilder sb = new StringBuilder();

while (!stack.isEmpty()) {

sb.append(stack.pop());

}

String result = sb.reverse().toString();

System.out.println(result);

}

}

Example

input: "abbaca"

Expected output: "ca"

Program 10:

EVALUATE THE VALUE OF AN ARITHMETIC EXPRESSION IN REVERSE POLISH NOTATION (RPN). VALID OPERATORS ARE +, -, \*, AND /. EACH OPERAND MAY BE AN INTEGER OR ANOTHER EXPRESSION.

Solution

import java.util.\*;

public class {

public static void main(String[] args) {

int n=sc.nextInt();

String[] st=new String[n];

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

St[i]=sc.next();

}

int result = method(st);

System.out.println(result);

}

public static int method(String[] st) {

Stack<Integer> stack = new Stack<>();

for (String n :st) {

if (n.equals("+")) {

int a = stack.pop();

int b = stack.pop();

stack.push(b + a);

} else if (n.equals("-")) {

int a = stack.pop();

int b = stack.pop();

stack.push(b - a);

} else if (n.equals("\*")) {

int a = stack.pop();

int b = stack.pop();

stack.push(b \* a);

} else if (n.equals("/")) {

int a = stack.pop();

int b = stack.pop();

stack.push(b / a);

} else {

stack.push(Integer.parseInt(n));

}

}

return stack.pop();

}

}

Example

input: ["2", "1", "+", "3", "\*"]

Expected output: 9