**Laboratory work #4.**

Solve these problems using stack, queue, deque data structures.

Deadline: week 5

<https://informatics.msk.ru/mod/statements/view.php?id=207#1>

package com.company;  
import java.util.Scanner;  
  
class Main {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 Stack st = new Stack();  
 int x;  
 String s;  
 while (true) {  
 s = scanner.next();  
 if (s.equals("push")) {  
 x = scanner.nextInt();  
 st.push(x);  
 } else if (s.equals("pop"))  
 st.pop();  
 else if (s.equals("back"))  
 st.back();  
 else if (s.equals("size"))  
 st.size();  
 else if (s.equals("clear"))  
 st.clear();  
 else {  
 st.exit();  
 return;  
 }  
 }  
 }  
 static class Stack{  
 private int mSize;  
 private int[] stackArray = new int[10000];  
 public void push(int element) {  
 stackArray[mSize++] = element;  
 System.*out*.println("ok");  
 }  
 public void pop() {  
 System.*out*.println(stackArray[mSize - 1]);  
 --mSize;  
 }  
 public void back(){  
 System.*out*.println(stackArray[mSize - 1]);  
 }  
 public void size() {  
 System.*out*.println(mSize);  
 }  
 public void clear() {  
 mSize = 0;  
 System.*out*.println("ok");  
 }  
 public void exit() {  
 System.*out*.println("bye");  
 }  
 }  
}

<https://informatics.msk.ru/mod/statements/view.php?id=207&chapterid=55#1>

package Task2;  
import java.util.Scanner;  
  
class Main {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 Stack st = new Stack();  
 int x;  
 String s;  
 while (true) {  
 s = scanner.next();  
 if (s.equals("push")) {  
 x = scanner.nextInt();  
 st.push(x);  
 }else if (s.equals("size"))  
 st.size();  
 else {  
 st.exit();  
 return;  
 }  
 }  
 }  
 static class Stack{  
 private int mSize;  
 private int[] stackArray = new int[10000];  
 public void push(int element) {  
 stackArray[mSize++] = element;  
 System.*out*.println("ok");  
 }  
 public void size() {  
 System.*out*.println(mSize);  
 }  
 public void exit() {  
 System.*out*.println("bye");  
 }  
 }  
}

<https://informatics.msk.ru/mod/statements/view.php?id=207&chapterid=57#1>

package TAsk3;  
import java.util.Scanner;  
  
class Main {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 Queue queue = new Queue();  
 int x;  
 String s;  
 while (true) {  
 s = scanner.next();  
 if (s.equals("push")) {  
 x = scanner.nextInt();  
 queue.push(x);  
 }else if (s.equals("size"))  
 queue.size();  
 else {  
 queue.exit();  
 return;  
 }  
 }  
 }  
 static class Queue{  
 private int mSize;  
 private int[] queueArray = new int[10000];  
 public void push(int element) {  
 queueArray[mSize++] = element;  
 System.*out*.println("ok");  
 }  
 public void size() {  
 System.*out*.println(mSize);  
 }  
 public void exit() {  
 System.*out*.println("bye");  
 }  
 }  
}

<https://informatics.msk.ru/mod/statements/view.php?id=207&chapterid=58#1>

package Task4;  
import java.util.Scanner;  
  
class Main {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 Queue queue = new Queue();  
 int x;  
 String s;  
 while (true) {  
 s = scanner.next();  
 if (s.equals("push")) {  
 x = scanner.nextInt();  
 queue.push(x);  
 }else if (s.equals("pop"))  
 queue.pop();  
 else if (s.equals("front"))  
 queue.front();  
 else if (s.equals("size"))  
 queue.size();  
 else if (s.equals("clear"))  
 queue.clear();  
 else {  
 queue.exit();  
 return;  
 }  
 }  
 }  
 static class Queue{  
 private int mSize;  
 private int rear = 0;  
 private int[] queueArray = new int[10000];  
 public void push(int element) {  
 queueArray[mSize++] = element;  
 System.*out*.println("ok");  
 rear++;  
 }  
 public void pop() {  
 System.*out*.println(queueArray[0]);  
 --mSize;  
 }  
 public void front(){  
 System.*out*.println(queueArray[0]);  
 }  
 public void size() {  
 System.*out*.println(mSize);  
 }  
 public void clear() {  
 mSize = 0;  
 System.*out*.println("ok");  
 }  
 public void exit() {  
 System.*out*.println("bye");  
 }  
 }  
}

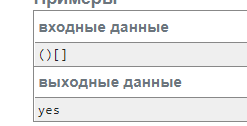
<https://informatics.msk.ru/mod/statements/view.php?id=207&chapterid=60#1>

public class Main{  
 static public void main (String[] args) throws IOException {  
 Deque dq = new Deque();  
 BufferedReader br = new BufferedReader(new InputStreamReader(System.*in*));  
 String str;  
  
 while((str = br.readLine()) != null) {  
 String[] inp = str.split(" ");  
 if (inp[0].equals("push\_front")){  
 dq.push\_front(Integer.*parseInt*(inp[1]));  
 } else if (inp[0].equals("push\_back")) {  
 dq.push\_back(Integer.*parseInt*(inp[1]));  
 } else if (inp[0].equals("pop\_back")) {  
 System.*out*.println(dq.pop\_back());  
 } else if (inp[0].equals("pop\_front")) {  
 System.*out*.println(dq.pop\_front());  
 } else if (inp[0].equals("front")) {  
 System.*out*.println(dq.front());  
 } else if (inp[0].equals("back")) {  
 System.*out*.println(dq.back());  
 } else if (inp[0].equals("size")) {  
 System.*out*.println(dq.size());  
 } else if (inp[0].equals("clear")) {  
 dq.clear();  
 } else if (inp[0].equals("exit")) {  
 System.*out*.println("bye");  
 br.close();  
 return;  
 }  
 }  
 }  
}  
  
class Deque {  
 final int maxSize = 10000;  
 int size;  
 public int head, end;  
 int[] data;  
  
 public Deque() {  
 data = new int[maxSize];  
 head = 0;  
 end = 0;  
 size = 0;  
 }  
  
 public void push\_front(int e) {  
 if (size == maxSize) {  
 System.*out*.println("Deque is full.");  
 return;  
 } else if (size==0) {  
 end = head;  
 data[head] = e;  
 size++;  
 } else {  
 head++;  
 if (head>=maxSize)  
 head = 0;  
 data[head] = e;  
 size++;  
 }  
 System.*out*.println("ok");  
 }  
  
 public void push\_back(int e) {  
 if (size == maxSize) {  
 System.*out*.println("Deque is full.");  
 return;  
 } else if (size==0) {  
 head = end;  
 data[end] = e;  
 size++;  
 } else {  
 end--;  
 if (end<0)  
 end = maxSize-1;  
 data[end] = e;  
 size++;  
 }  
 System.*out*.println("ok");  
 }  
  
 public void clear() {  
 head = 0;  
 end = 0;  
 size = 0;  
 System.*out*.println("ok");  
 }  
  
 public int size() {  
 return size;  
 }  
  
 public int back() {  
 if (size!=0)  
 return data[end];  
 else  
 return -1;  
 }  
  
 public int front() {  
 if (size!=0)  
 return data[head];  
 else  
 return -1;  
 }  
  
 public int pop\_back() {  
 if (size!=0) {  
 int tmp = data[end];  
 end++;  
 if (end>=maxSize)  
 end = 0;  
 size --;  
 return tmp;  
 }  
 else  
 return -1;  
 }  
  
 public int pop\_front() {  
 if (size!=0) {  
 int tmp = data[head];  
 head--;  
 if (head<0)  
 head = maxSize - 1;  
 size --;  
 return tmp;  
 }  
 else  
 return -1;  
 }  
}

<https://informatics.msk.ru/mod/statements/view.php?id=206&chapterid=50#1>

public class Main {  
  
 public static void main(String[] args) {  
 int[] a1 = {1, 3, 5, 7, 9};  
 int[] a2 = {2, 4, 6, 8, 0};  
 ArrayList<Integer> arr1 = new ArrayList<>();  
 ArrayList<Integer> arr2 = new ArrayList<>();  
 for (int i = 0; i < 5; i++) {  
 arr1.add(a1[i]);  
 arr2.add(a2[i]);  
 }  
  
 int count = 0;  
 boolean res = false;  
  
 for (int i = 0; i < 106; i++) {  
 count++;  
 if (arr1.get(0) > arr2.get(0) || ((arr1.get(0) == 0) && (arr2.get(0) == 9))) {  
 arr1.add(arr1.get(0));  
 arr1.add(arr2.get(0));  
 } else if (arr1.get(0) < arr2.get(0) || ((arr1.get(0) == 9) && (arr2.get(0) == 0))) {  
 arr2.add(arr2.get(0));  
 arr2.add(arr1.get(0));  
 }  
 arr1.remove(0);  
 arr2.remove(0);  
  
 if (arr1.size() == 0) {  
 System.*out*.println("Second " + count);  
 res = true;  
 break;  
 } else if (arr2.size() == 0) {  
 System.*out*.println("First " + count);  
 res = true;  
 break;  
 }  
 }  
 if (!res) System.*out*.println("Botva!");  
 }  
}

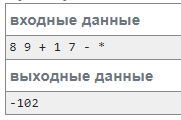
<https://informatics.msk.ru/mod/statements/view.php?id=206&chapterid=51#1>



package Task7;  
  
import java.util.Stack;  
import java.util.Scanner;  
  
public class Main {  
  
 public static void main(String[] args) {  
 Stack<Integer> st = new Stack<>();  
 Scanner inp = new Scanner(System.*in*);  
 System.*out*.print("cmd=");  
 String cmd = inp.nextLine();  
 char a;  
 int i,k;  
 boolean flag=true;  
 for (i=0; i<cmd.length(); i++){  
 a=cmd.charAt(i);  
 switch (a){  
 case '(':  
 st.push(1);  
 break;  
 case '[':  
 st.push(2);  
 break;  
 case '{':  
 st.push(3);  
 break;  
 case ')':  
 if (st.empty()) {  
 flag=false;  
 break;  
 }  
 k=st.peek();  
 st.pop();  
 if (k!=1) flag=false;  
 break;  
 case ']':  
 if (st.empty()) {  
 flag=false;  
 break;  
 }  
 k=st.peek();  
 st.pop();  
 if (k!=2) flag=false;  
 break;  
 case '}':  
 if (st.empty()) {  
 flag=false;  
 break;  
 }  
 k=st.peek();  
 st.pop();  
 if (k!=3) flag=false;  
 break;  
 default:  
 }  
 if (! flag) break;  
 }  
  
 if (flag && st.empty())  
 System.*out*.println("OK");  
 else  
 System.*out*.println("Error");  
 }  
}

<https://informatics.msk.ru/mod/statements/view.php?id=206&chapterid=52#1>

Post fix notation



class Solution {

public int evalRPN(String[] tokens) {

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

for(String c:tokens){

if(!c.equals("+") && !c.equals("-") && !c.equals("\*") && !c.equals("/")){

s.push(Integer.parseInt(c));

}else{

int a = s.pop();

int b = s.pop();

if(c.equals("+")) s.push(b+a);

else if (c.equals("-")) s.push(b-a);

else if(c.equals("\*")) s.push(b\*a);

else s.push(b/a);

}

}

return s.pop();

}

}

<https://leetcode.com/problems/valid-parentheses/>

class Solution {

public boolean isValid(String s) {

if(s.length() < 2){

return false;

}

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

char[] c = s.toCharArray();

for(char x : c){

if(stack.isEmpty()){

stack.push(x);

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

//open

stack.push(x);

}else{

char check = stack.peek();

if(x == ')' && check == '('){

stack.pop();

}else if(x == ']' && check == '['){

stack.pop();

}else if(x == '}' && check == '{'){

stack.pop();

}else{

return false;

}

}

}

if(stack.isEmpty()){

return true;

}else{

return false;

}

}

}

<https://leetcode.com/problems/min-stack/>

class MinStack {

Stack<Integer> stack;

int min;

public MinStack() {

min = Integer.MAX\_VALUE;

stack = new Stack<>();

}

public void push(int x) {

if(x <= min){

stack.push(min);

min = x;

}

stack.push(x);

}

public void pop() {

int x = stack.pop();

if(x == min){

min = stack.pop();

}

}

public int top() {

return stack.peek();

}

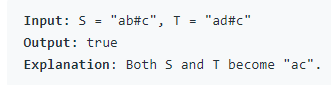
public int getMin() {

return min;

}

}

<https://leetcode.com/problems/backspace-string-compare/>



class Solution {

public boolean backspaceCompare(String S, String T) {

return build(S).equals(build(T));

}

public String build(String S){

Stack<Character> ans = new Stack();

for(char c: S.toCharArray()){

if(c != '#')

ans.push(c);

else if (!ans.empty())

ans.pop();

}

return String.valueOf(ans); // convert stact to string

}

}

<https://leetcode.com/problems/evaluate-reverse-polish-notation/>

class Solution {

public int evalRPN(String[] tokens) {

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

for(String c:tokens){

if(!c.equals("+") && !c.equals("-") && !c.equals("\*") && !c.equals("/")){

s.push(Integer.parseInt(c));

}else{

int a = s.pop();

int b = s.pop();

int ans;

if(c.equals("+")) s.push(b+a);

else if (c.equals("-")) s.push(b-a);

else if(c.equals("\*")) s.push(b\*a);

else s.push(b/a);

}

}

return s.pop();

}

}