Code : 1  
package Topic\_14\_Stacks;

import java.util.Scanner;

import java.util.Stack;

public class A\_IsDuplicateBrackets {

public static void main(String[] args) {

Scanner s = new Scanner(System.in);

String ip = s.nextLine();

boolean res = isDuplicateBrackets("(a + b) + (c + d)");

System.out.println(res);

res = isBalancedBrackets("(a+b)+(c+d)");

System.out.println(res);

}

private static boolean isDuplicateBrackets(String ip) {

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

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

char ch = ip.charAt(i);

if (ch == ')') {

if (s.peek() == '(') {

return true;

} else {

while (s.peek() != '(') {

s.pop();

}

s.pop();

}

} else {

s.push(ch);

}

}

return false;

}

private static boolean isBalancedBrackets(String ip) {

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

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

char ch = ip.charAt(i);

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

s.push(ch);

} else if (ch == ')' || ch == ']' || ch == '}') {

if (!s.isEmpty()) s.pop();

}

}

if (s.size() > 0) {

return true;

}

return false;

}

}

Code : 2  
package Topic\_14\_Stacks;

import java.util.Scanner;

import java.util.Stack;

public class B\_IsBalancedBrackets {

public static void main(String[] args) throws Exception {

Scanner scn = new Scanner(System.in);

String str = scn.nextLine();

boolean res = IsBracketsBalanced(str);

System.out.println(res);

}

private static boolean IsBracketsBalanced(String str) {

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

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

char ch = str.charAt(i);

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

st.push(ch);

} else if (ch == ')') {

boolean val = handleClosing(st, '(');

if (val == false) {

return false;

}

} else if (ch == ']') {

boolean val = handleClosing(st, '[');

if (val == false) {

return false;

}

} else if (ch == '}') {

boolean val = handleClosing(st, '{');

if (val == false) {

return false;

}

}

}

if (st.size() == 0) {

return true;

} else {

return false;

}

}

public static boolean handleClosing(Stack<Character> st, char corresopch) {

if (st.size() == 0) {

return false;

} else if (st.peek() != corresopch) {

return false;

} else {

st.pop();

return true;

}

}

}

Code : 3  
package Topic\_14\_Stacks;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.util.Stack;

public class C\_NextGreaterElementToTheRight {

public static void display(int[] a) {

StringBuilder sb = new StringBuilder();

for (int val : a) {

sb.append(val + "\n");

}

System.out.println(sb);

}

public static void main(String[] args) throws Exception {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

int n = Integer.parseInt(br.readLine());

int[] a = new int[n];

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

a[i] = Integer.parseInt(br.readLine());

}

int[] nge = solve(a);

display(nge);

}

public static int[] solve(int[] arr) {

int[] nge = new int[arr.length];

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

int arrLength = arr.length - 1;

nge[arrLength] = -1;

st.push(arr[arrLength]);

for (int i = arr.length - 2; i >= 0; i--) {

while (st.size() > 0 && arr[i] >= st.peek()) {

st.pop();

}

if (st.size() == 0) {

nge[i] = -1;

} else {

nge[i] = st.peek();

}

st.push(arr[i]);

}

return nge;

}

}

Code : 4  
package Topic\_14\_Stacks;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.util.Stack;

public class D\_StockSpan {

public static void display(int[] a) {

StringBuilder sb = new StringBuilder();

for (int val : a) {

sb.append(val + "\n");

}

System.out.println(sb);

}

public static void main(String[] args) throws Exception {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

int n = Integer.parseInt(br.readLine());

int[] a = new int[n];

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

a[i] = Integer.parseInt(br.readLine());

}

int[] span = solve(a);

display(span);

}

public static int[] solve(int[] arr) {

int[] span = new int[arr.length];

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

st.push(0);

span[0] = 1;

for (int i = 1; i < arr.length; i++) {

while (st.size() > 0 && arr[i] >= arr[st.peek()]) {

st.pop();

}

if (st.size() == 0) {

span[i] = i + 1;

} else {

span[i] = i - st.peek();

}

st.push(i);

}

return span;

}

}

Code : 5  
package Topic\_14\_Stacks;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.util.Stack;

public class E\_LargestAreaHistogram {

public static void main(String[] args) throws Exception {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

int n = Integer.parseInt(br.readLine());

int[] a = new int[n];

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

a[i] = Integer.parseInt(br.readLine());

}

// code

int[] rb = new int[n]; // nse index to right

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

st.push(n - 1);

rb[n - 1] = n;

for (int i = n - 2; i >= 0; i--) {

while (st.size() > 0 && a[i] <= a[st.peek()]) {

st.pop();

}

if (st.size() > 0) {

rb[i] = st.peek();

} else {

rb[i] = n;

}

st.push(i);

}

int[] lb = new int[n]; // nse index to left

st = new Stack<>();

st.push(0);

lb[0] = -1;

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

while (st.size() > 0 && a[i] <= a[st.peek()]) {

st.pop();

}

if (st.size() > 0) {

lb[i] = st.peek();

} else {

lb[i] = -1;

}

st.push(i);

}

int maxArea = 0;

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

int width = rb[i] - lb[i] - 1;

int area = a[i] \* width;

if (area > maxArea) {

maxArea = area;

}

}

System.out.println(maxArea);

}

}

Code : 6  
package Topic\_14\_Stacks;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.util.Stack;

public class F\_SlidingWindow {

public static void main(String[] args) throws Exception {

BufferedReader br = new BufferedReader(new

InputStreamReader(System.in));

int n = Integer.parseInt(br.readLine());

int[] arr = new int[n];

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

arr[i] = Integer.parseInt(br.readLine());

}

int k = Integer.parseInt(br.readLine());

slidingWindow(arr, k);

}

private static void slidingWindow(int[] arr, int k) {

// nge begin

int[] nge = new int[arr.length];

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

st.push(arr.length - 1);

nge[arr.length - 1] = arr.length;

for (int i = arr.length - 2; i >= 0; i--) {

while (st.size() > 0 && arr[i] >= arr[st.peek()]) {

st.pop();

}

if (st.size() == 0) {

nge[i] = arr.length;

} else {

nge[i] = st.peek();

}

st.push(i);

}

// nge end

int i = 0;

for (int w = 0; w <= arr.length - k; w++) {

if (i < w) {

i = w;

}

while (nge[i] < w + k) {

i = nge[i];

}

System.out.println(arr[i]);

}

}

}

Code : 7  
package Topic\_14\_Stacks;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.util.Stack;

public class G\_InfixEvaluation {

public static int evaluate(String expression) {

char[] tokens = expression.toCharArray();

// Stack for numbers: 'values'

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

// Stack for Operators: 'ops'

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

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

// Current token is a

// whitespace, skip it

if (tokens[i] == ' ') continue;

// Current token is a number,

// push it to stack for numbers

if (tokens[i] >= '0' && tokens[i] <= '9') {

StringBuilder sbuf = new StringBuilder();

// There may be more than one

// digits in number

while (i < tokens.length && tokens[i] >= '0' && tokens[i] <= '9') sbuf.append(tokens[i++]);

values.push(Integer.parseInt(sbuf.toString()));

// right now the i points to

// the character next to the digit,

// since the for loop also increases

// the i, we would skip one

// token position; we need to

// decrease the value of i by 1 to

// correct the offset.

i--;

}

// Current token is an opening brace,

// push it to 'ops'

else if (tokens[i] == '(') ops.push(tokens[i]);

// Closing brace encountered,

// solve entire brace

else if (tokens[i] == ')') {

while (ops.peek() != '(') values.push(applyOp(ops.pop(), values.pop(), values.pop()));

ops.pop();

}

// Current token is an operator.

else if (tokens[i] == '+' || tokens[i] == '-' || tokens[i] == '\*' || tokens[i] == '/') {

// While top of 'ops' has same

// or greater precedence to current

// token, which is an operator.

// Apply operator on top of 'ops'

// to top two elements in values stack

while (!ops.empty() && hasPrecedence(tokens[i], ops.peek()))

values.push(applyOp(ops.pop(), values.pop(), values.pop()));

// Push current token to 'ops'.

ops.push(tokens[i]);

}

}

// Entire expression has been

// parsed at this point, apply remaining

// ops to remaining values

while (!ops.empty()) values.push(applyOp(ops.pop(), values.pop(), values.pop()));

// Top of 'values' contains

// result, return it

return values.pop();

}

// Returns true if 'op2' has higher

// or same precedence as 'op1',

// otherwise returns false.

public static boolean hasPrecedence(char op1, char op2) {

if (op2 == '(' || op2 == ')') return false;

if ((op1 == '\*' || op1 == '/') && (op2 == '+' || op2 == '-')) return false;

else return true;

}

// A utility method to apply an

// operator 'op' on operands 'a'

// and 'b'. Return the result.

public static int applyOp(char op, int b, int a) {

switch (op) {

case '+':

return a + b;

case '-':

return a - b;

case '\*':

return a \* b;

case '/':

if (b == 0) throw new UnsupportedOperationException("Cannot divide by zero");

return a / b;

}

return 0;

}

// Driver method to test above methods

public static void main(String[] args) throws Exception {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String exp = br.readLine();

int evaluate = evaluate(exp);

System.out.println(evaluate);

// code

}

}

Code : 8  
package Topic\_14\_Stacks;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.util.Stack;

public class H\_InfixConversion {

public static void main(String[] args) throws Exception {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String exp = br.readLine();

// code

Stack<String> postfix = new Stack<>();

Stack<String> prefix = new Stack<>();

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

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

char ch = exp.charAt(i);

if (ch == '(') {

operators.push(ch);

} else if ((ch >= '0' && ch <= '9') || (ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z')) {

postfix.push(ch + "");

prefix.push(ch + "");

} else if (ch == '+' || ch == '-' || ch == '\*' || ch == '/') {

while (operators.size() > 0 && operators.peek() != '(' && precedence(ch) <= precedence(operators.peek())) {

char op = operators.pop();

String postval2 = postfix.pop();

String postval1 = postfix.pop();

postfix.push(postval1 + postval2 + op);

String preval2 = prefix.pop();

String preval1 = prefix.pop();

prefix.push(op + preval1 + preval2);

}

operators.push(ch);

} else if (ch == ')') {

while (operators.size() > 0 && operators.peek() != '(') {

char op = operators.pop();

String postval2 = postfix.pop();

String postval1 = postfix.pop();

postfix.push(postval1 + postval2 + op);

String preval2 = prefix.pop();

String preval1 = prefix.pop();

prefix.push(op + preval1 + preval2);

}

if (operators.size() > 0) {

operators.pop();

}

}

}

while (operators.size() > 0) {

char op = operators.pop();

String postval2 = postfix.pop();

String postval1 = postfix.pop();

postfix.push(postval1 + postval2 + op);

String preval2 = prefix.pop();

String preval1 = prefix.pop();

prefix.push(op + preval1 + preval2);

}

System.out.println(postfix.peek());

System.out.println(prefix.peek());

}

public static int precedence(char op) {

if (op == '+') {

return 1;

} else if (op == '-') {

return 1;

} else if (op == '\*') {

return 2;

} else {

return 2;

}

}

}

Code : 9  
package Topic\_14\_Stacks;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.util.Stack;

public class I\_PostfixEvaluationAndConversion {

public static void main(String[] args) throws Exception {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String exp = br.readLine();

Stack<Integer> vs = new Stack<>(); //1

Stack<String> is = new Stack<>();

Stack<String> ps = new Stack<>();

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

char ch = exp.charAt(i); //2

if (ch == '+' || ch == '-' || ch == '\*' || ch == '/') { //3

int v2 = vs.pop();

int v1 = vs.pop();

int val = operation(v1, v2, ch);

vs.push(val);

String iv2 = is.pop(); //4

String iv1 = is.pop();

String ival = "(" + iv1 + ch + iv2 + ")";

is.push(ival);

String pv2 = ps.pop(); //5

String pv1 = ps.pop();

String pval = ch + pv1 + pv2;

ps.push(pval);

} else {

vs.push(ch - '0'); //6

is.push(ch + "");

ps.push(ch + "");

}

}

System.out.println(vs.pop()); //7

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

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

}

public static int operation(int v1, int v2, char op) { //8

if (op == '+') {

return v1 + v2;

} else if (op == '-') {

return v1 - v2;

} else if (op == '\*') {

return v1 \* v2;

} else {

return v1 / v2;

}

}

}

Code : 10  
package Topic\_14\_Stacks;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.util.Stack;

public class I\_PrefixEvaluationAndConversion {

public static void main(String[] args) throws Exception {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String exp = br.readLine();

Stack<Integer> vs = new Stack<>(); //1

Stack<String> is = new Stack<>();

Stack<String> ps = new Stack<>();

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

char ch = exp.charAt(i); //2

if (ch == '+' || ch == '-' || ch == '\*' || ch == '/') { //3

int v1 = vs.pop();

int v2 = vs.pop();

int val = operation(v1, v2, ch);

vs.push(val);

String iv1 = is.pop(); //4

String iv2 = is.pop();

String ival = "(" + iv1 + ch + iv2 + ")";

is.push(ival);

String pv1 = ps.pop(); //5

String pv2 = ps.pop();

String pval = pv1 + pv2 + ch;

ps.push(pval);

} else {

vs.push(ch - '0'); //6

is.push(ch + "");

ps.push(ch + "");

}

}

System.out.println(vs.pop()); //7

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

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

}

public static int operation(int v1, int v2, char op) { //8

if (op == '+') {

return v1 + v2;

} else if (op == '-') {

return v1 - v2;

} else if (op == '\*') {

return v1 \* v2;

} else {

return v1 / v2;

}

}

}

Code : 11  
package Topic\_14\_Stacks;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.util.Stack;

public class J\_CelebrityProblem {

public static void main(String[] args) throws Exception {

// write your code here

BufferedReader br = new BufferedReader(new

InputStreamReader(System.in));

int n = Integer.parseInt(br.readLine());

int[][] arr = new int[n][n];

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

String line = br.readLine();

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

arr[j][k] = line.charAt(k) - '0';

}

}

findCelebrity(arr);

}

public static void findCelebrity(int[][] arr) {

// if a celebrity is there print it's index (not position),

// if there is not then print "none"

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

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

st.push(i);

}

while (st.size() > 1) {

int i = st.pop();

int j = st.pop();

if (arr[i][j] == 1) {

st.push(j);

} else {

st.push(i);

}

}

int pot = st.pop();

boolean flag = true;

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

if (i != pot) {

if (arr[i][pot] == 0 || arr[pot][i] == 1) {

flag = false;

break;

}

}

}

if (flag) {

System.out.println(pot);

} else {

System.out.println("none");

}

}

}

Code : 12  
package Topic\_14\_Stacks;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.util.Arrays;

import java.util.Stack;

public class K\_MergeOverlappingTimes {

public static void main(String[] args) throws Exception {

// write your code here

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

int n = Integer.parseInt(br.readLine());

int[][] arr = new int[n][2];

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

String line = br.readLine();

arr[j][0] = Integer.parseInt(line.split(" ")[0]);

arr[j][1] = Integer.parseInt(line.split(" ")[1]);

}

mergeOverlappingIntervals(arr);

}

public static void mergeOverlappingIntervals(int[][] arr) {

Pair[] pairs = new Pair[arr.length];

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

pairs[i] = new Pair(arr[i][0], arr[i][1]);

}

Arrays.sort(pairs);

Stack<Pair> st = new Stack<>();

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

if (i == 0) {

st.push(pairs[i]);

} else {

Pair top = st.peek();

if (pairs[i].st > top.et) {

st.push(pairs[i]);

} else {

top.et = Math.max(top.et, pairs[i].et);

}

}

}

Stack<Pair> rs = new Stack<>();

while (st.size() > 0) {

rs.push(st.pop());

}

while (rs.size() > 0) {

Pair p = rs.pop();

System.out.println(p.st + " " + p.et);

}

}

public static class Pair implements Comparable<Pair> {

int st;

int et;

Pair(int st, int et) {

this.st = st;

this.et = et;

}

public int compareTo(Pair other) {

if (this.st != other.st) {

return this.st - other.st;

} else {

return this.et - other.et;

}

}

}

}

Code : 13  
package Topic\_14\_Stacks;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.util.Stack;

public class L\_SmallestNumberFollowingPattern {

public static void main(String[] args) throws Exception {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String str = br.readLine();

// code

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

int num = 1;

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

char ch = str.charAt(i);

if (ch == 'd') { // when we encounter d

st.push(num);

num++;

} else { // when we encounter i

st.push(num);

num++;

while (st.size() > 0) {

System.out.print(st.pop());

}

}

}

st.push(num); // for last number

while (st.size() > 0) {

System.out.print(st.pop());

}

}

}

Code : 14  
package Topic\_14\_Stacks;

import java.io.BufferedReader;

import java.io.InputStreamReader;

public class M\_NormalStack {

public static class CustomStack {

int[] data;

int tos;

public CustomStack(int cap) {

data = new int[cap];

tos = -1;

}

int size() {

return tos + 1;

}

void display() {

for (int i = tos; i >= 0; i--) {

System.out.print(data[i] + " ");

}

System.out.println();

}

void push(int val) {

if (tos == data.length - 1) {

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

} else {

tos++;

data[tos] = val;

}

}

int pop() {

if (tos == -1) {

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

return -1;

} else {

int val = data[tos];

tos--;

return val;

}

}

int top() {

if (tos == -1) {

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

return -1;

} else {

return data[tos];

}

}

}

public static void main(String[] args) throws Exception {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

int n = Integer.parseInt(br.readLine());

CustomStack st = new CustomStack(n);

String str = br.readLine();

while (str.equals("quit") == false) {

if (str.startsWith("push")) {

int val = Integer.parseInt(str.split(" ")[1]);

st.push(val);

} else if (str.startsWith("pop")) {

int val = st.pop();

if (val != -1) {

System.out.println(val);

}

} else if (str.startsWith("top")) {

int val = st.top();

if (val != -1) {

System.out.println(val);

}

} else if (str.startsWith("size")) {

System.out.println(st.size());

} else if (str.startsWith("display")) {

st.display();

}

str = br.readLine();

}

}

}

Code : 15  
package Topic\_14\_Stacks;

import java.io.BufferedReader;

import java.io.InputStreamReader;

public class N\_DynamicStack {

public static class CustomStack {

int[] data;

int tos;

public CustomStack(int cap) {

data = new int[cap];

tos = -1;

}

int size() {

return tos + 1;

}

void display() {

for (int i = tos; i >= 0; i--) {

System.out.print(data[i] + " ");

}

System.out.println();

}

void push(int val) {

if (tos == data.length - 1) {

int[] ndata = new int[2 \* data.length];

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

ndata[i] = data[i];

}

data = ndata;

}

tos++;

data[tos] = val;

}

int pop() {

if (tos == -1) {

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

return -1;

} else {

int val = data[tos];

tos--;

return val;

}

}

int top() {

if (tos == -1) {

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

return -1;

} else {

return data[tos];

}

}

}

public static void main(String[] args) throws Exception {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

int n = Integer.parseInt(br.readLine());

CustomStack st = new CustomStack(n);

String str = br.readLine();

while (str.equals("quit") == false) {

if (str.startsWith("push")) {

int val = Integer.parseInt(str.split(" ")[1]);

st.push(val);

} else if (str.startsWith("pop")) {

int val = st.pop();

if (val != -1) {

System.out.println(val);

}

} else if (str.startsWith("top")) {

int val = st.top();

if (val != -1) {

System.out.println(val);

}

} else if (str.startsWith("size")) {

System.out.println(st.size());

} else if (str.startsWith("display")) {

st.display();

}

str = br.readLine();

}

}

}

Code : 16  
package Topic\_14\_Stacks;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.util.Stack;

public class O\_MinStack1 {

public static class MinStack {

Stack<Integer> allData;

Stack<Integer> minData;

public MinStack() {

allData = new Stack<>();

minData = new Stack<>();

}

int size() {

return allData.size();

}

void push(int val) {

allData.push(val);

if (minData.size() == 0 || val <= minData.peek()) {

minData.push(val);

}

}

int pop() {

if (size() == 0) {

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

return -1;

} else {

int val = allData.pop();

if (val == minData.peek()) {

minData.pop();

}

return val;

}

}

int top() {

if (size() == 0) {

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

return -1;

} else {

return allData.peek();

}

}

int min() {

if (size() == 0) {

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

return -1;

} else {

return minData.peek();

}

}

}

public static void main(String[] args) throws Exception {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

MinStack st = new MinStack();

String str = br.readLine();

while (str.equals("quit") == false) {

if (str.startsWith("push")) {

int val = Integer.parseInt(str.split(" ")[1]);

st.push(val);

} else if (str.startsWith("pop")) {

int val = st.pop();

if (val != -1) {

System.out.println(val);

}

} else if (str.startsWith("top")) {

int val = st.top();

if (val != -1) {

System.out.println(val);

}

} else if (str.startsWith("size")) {

System.out.println(st.size());

} else if (str.startsWith("min")) {

int val = st.min();

if (val != -1) {

System.out.println(val);

}

}

str = br.readLine();

}

}

}

Code : 17  
package Topic\_14\_Stacks;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.util.Stack;

public class P\_MinStackConstantSpace {

public static class MinStack {

Stack<Integer> data;

int min;

public MinStack() {

data = new Stack<>();

}

int size() {

return data.size();

}

void push(int val) {

if (size() == 0) {

data.push(val);

min = val;

} else if (val < min) {

data.push(val + val - min);

min = val;

} else {

data.push(val);

}

}

int pop() {

if (size() == 0) {

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

return -1;

} else {

if (data.peek() < min) {

int oval = min;

min = 2 \* min - data.pop();

return oval;

} else {

return data.pop();

}

}

}

int top() {

if (size() == 0) {

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

return -1;

} else {

if (data.peek() >= min) {

return data.peek();

} else {

return min;

}

}

}

int min() {

if (size() == 0) {

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

return -1;

} else return min;

}

}

public static void main(String[] args) throws Exception {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

MinStack st = new MinStack();

String str = br.readLine();

while (str.equals("quit") == false) {

if (str.startsWith("push")) {

int val = Integer.parseInt(str.split(" ")[1]);

st.push(val);

} else if (str.startsWith("pop")) {

int val = st.pop();

if (val != -1) {

System.out.println(val);

}

} else if (str.startsWith("top")) {

int val = st.top();

if (val != -1) {

System.out.println(val);

}

} else if (str.startsWith("size")) {

System.out.println(st.size());

} else if (str.startsWith("min")) {

int val = st.min();

if (val != -1) {

System.out.println(val);

}

}

str = br.readLine();

}

}

}

Code : 18  
package Topic\_14\_Stacks;

public class R\_BaseBallGame {

}

Code : 19  
package Topic\_14\_Stacks;

public class S\_MiniParser {

}