//Chris Clemmons

//CSC310

// Professor Kim

// 2/13/19

package christopher\_clemmons.hw3;

import java.util.\*;

class MyStack {

private double[] stack; //creates the int stack object

private char[] stack2;

private int size = 10; // sets size of stack

private int size2 = 10;

private int position = -1; // sets current position of stack

private int position2 = -1;

void CreateDoubleStack(){ // creates the stack with size

stack = new double[size];

}

void CreateCharStack(){ // creates the stack with size

stack2 = new char[size];

}

public double MyDoublePush(double pushDouble){ // pushes element into stack

position++;

if (position ==stack.length){ // if stack is too small, increase stack size

IncreaseDoubleSize();

}

stack[position] = pushDouble; //adds double to stack

return pushDouble;

}

public char MyCharPush(char pushChar){ // pushes element into stack

position2++;

if (position2 ==stack2.length){

IncreaseCharSize(); //if stack is too small, increase stack size

}

stack2[position2] = pushChar; // pushes char to stack

return pushChar;

}

public double MyDoublePop(){ // removes double from stack

double p = stack[position];

position--;

return p;

}

public char MyCharPop(){ // removes char from stack

if(position2==-1){ // checks if

return 'n';

}

char p = stack2[position2];

position2--;

return p;

}

public double MyTop(){

double p = stack[position];

return p;

}

public char MyCharTop(){

if(position2==-1){

return 'n';

}

char p = stack2[position2];

return p;

}

public double GetMin(){

double min = stack[0];

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

if (min >= stack[i]){

min = stack[i];

}

}

return min;

}

private void IncreaseDoubleSize(){

int newSize = size+1;

size = newSize;

stack = Arrays.copyOf(stack, newSize);

}

private void IncreaseCharSize(){

int newSize = size2+1;

size2 = newSize;

stack2 = Arrays.copyOf(stack2, newSize);

}

public void evalExp(String e){

char [] exp = e.toCharArray();

MyStack doubleStack= new MyStack();

doubleStack.CreateDoubleStack();

MyStack charStack= new MyStack();

charStack.CreateCharStack();

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

if( exp[i]==' '){

continue;

}

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

String temp =""+ exp[i];

doubleStack.MyDoublePush(Double.parseDouble(temp));

}

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

charStack.MyCharPush(exp[i]);

}

if(exp[i]== ')'){

//charStack.MyCharPush(')');

//charStack.MyCharPop();

boolean char1 = true;

while(char1==true){

if(charStack.MyCharTop()=='('|| charStack.MyCharTop()=='n'){

;

charStack.MyCharPop();

char1=false;

continue;

}

char c = charStack.MyCharPop();

double x = doubleStack.MyDoublePop();

double y = doubleStack.MyDoublePop();

switch(c)

{

case '+':

doubleStack.MyDoublePush(y+x);

System.out.println(x+"+"+ y);

break;

case '-':

doubleStack.MyDoublePush(y-x);

System.out.println(y+"-"+x);

break;

case '/':

doubleStack.MyDoublePush(y/x);

System.out.println(y+"/"+x);

break;

case '\*':

doubleStack.MyDoublePush(x\*y);

System.out.println(y+"\*"+x);

break;

default:

break;

}

}

}

}

if(charStack.MyCharTop()!='n'){

boolean char1 = true;

while(char1==true){

if(charStack.MyCharTop()=='('|| charStack.MyCharTop()=='n'){

char1=false;

return;

}

char c = charStack.MyCharPop();

double x = doubleStack.MyDoublePop();

double y = doubleStack.MyDoublePop();

switch(c)

{

case '+':

doubleStack.MyDoublePush(y+x);

System.out.println(x+"+"+ y);

break;

case '-':

doubleStack.MyDoublePush(y-x);

System.out.println(y+"-"+x);

break;

case '/':

doubleStack.MyDoublePush(y/x);

System.out.println(y+"/"+x);

break;

case '\*':

doubleStack.MyDoublePush(x\*y);

System.out.println(y+"\*"+x);

break;

}

System.out.println("Output = "+doubleStack.MyDoublePop());

}

}

}

public void PostFix(String e){

char [] exp = e.toCharArray();

MyStack doubleStack= new MyStack();

MyStack charStack= new MyStack();

doubleStack.CreateDoubleStack();

int pos1=0;

int pos2=0;

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

char c = exp[i];

if( exp[i]==' '){

continue;

}

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

pos1++;

String temp =""+ exp[i];

int tempInt =Integer.parseInt(temp);

doubleStack.MyDoublePush(tempInt);

}

else

{

double x = doubleStack.MyDoublePop();

double y = doubleStack.MyDoublePop();

switch(c)

{

case '+':

doubleStack.MyDoublePush(y+x);

System.out.println(x+"+"+ y);

break;

case '-':

doubleStack.MyDoublePush(y-x);

System.out.println(y+"-"+x);

break;

case '/':

doubleStack.MyDoublePush(y/x);

System.out.println(y+"/"+x);

break;

case '\*':

doubleStack.MyDoublePush(x\*y);

System.out.println(y+"\*"+x);

break;

}

}

}

System.out.println("Output = " + doubleStack.MyDoublePop());

}

}

public class Christopher\_ClemmonsHW3 {

void postfix (String expression){

}

public static void main(String[] args) {

MyStack stack = new MyStack ();

stack.CreateDoubleStack();

//push the numbers to stack

System.out.println("Push: "+ stack.MyDoublePush(5));

System.out.println("Push: "+ stack.MyDoublePush(4));

System.out.println("Push: "+ stack.MyDoublePush(3));

System.out.println("Push: "+ stack.MyDoublePush(2));

System.out.println("Push: "+ stack.MyDoublePush(1));

//pop the numbers from stack

System.out.println("Pop: "+ stack.MyDoublePop());

//get top from stack

System.out.println("Top: "+ stack.MyTop());

//get min from stack

System.out.println("Min: "+ stack.GetMin());

//sets up strings for evalExp and PostFix

String exp = "((5+2)\*(8-3))/4";

String postFixExp = "52+83-\*4/";

//creates new stack to use with exp

MyStack stack2 = new MyStack();

stack2.evalExp(exp);

//creates new stack to use with PostFix

MyStack stack3 = new MyStack();

stack3.PostFix(postFixExp);

}

}