```
//Stack

class StackArray {

	int top;
	int stack[];
	static int CAPACITY = 100;

	public StackArray(){
		this.top = -1;
		this.stack = new int[CAPACITY];
	}

	public void push(int data){
		if(top == CAPACITY){
			System.out.println("Stack is full");
			return;
		}
		stack[++top] = data;
	}

	public int pop(){
		if(top == -1){
			System.out.println("Stack is empty");
			return -1;
		}
		int item = stack[top--];
		return item;
	}

	public boolean isEmpty(){
		if(top == -1)
			return true;
		return false;
	}

	public int size(){
		return (top+1);
	}

	public void printStack(){
		for(int i=0; i<=top; i++){
			System.out.println(stack[i] + " ");
		}
		System.out.println();
	}

	public int top(){
		return stack[top];
	}

	public static void main(String args[]){
		StackArray stackArr = new StackArray();
		stackArr.push(1);
		stackArr.push(2);
```

```
            stackArr.push(3);
            stackArr.push(4);
            stackArr.push(5);
            System.out.println("Size: " + stackArr.size());
            System.out.println("Top: " + stackArr.top());
            stackArr.printStack();

        }
}
```

```
import java.util.Stack;

class BalancedParenthesis {

    public boolean checkIfBalanced(String str){

        if(str == null || str.length() == 0)
            return true;
        Stack<Character> stack = new Stack<>();
        for(int i=0; i<str.length(); i++){
            if(str.charAt(i) == '(' || str.charAt(i) == '{' || str.charAt(i) == '[')
                stack.push(str.charAt(i));
            else if(str.charAt(i) == ')'){
                if(!stack.isEmpty() && stack.peek() == '(')
                    stack.pop();
                else
                    return false;
            }
            else if(str.charAt(i) == '}'){
                if(!stack.isEmpty() && stack.peek() == '{')
                    stack.pop();
                else
                    return false;
            }
            else if(str.charAt(i) == ']'){
                if(!stack.isEmpty() && stack.peek() == '[')
                    stack.pop();
                else
                    return false;
            }
        }
        if(stack.isEmpty())
            return true;
        else
            return false;
    }

    public int priority(char ch){
        if(ch == '+' || ch == '-')
            return 1;
        else if(ch == '*' || ch == '/')
            return 2;
        else if(ch == '^')
            return 3;
        return -1;
    }

    public String infixToPostfix(String str){
```

```
    if(str == null || str.length() == 0)
        return null;
    String result = "";
    Stack<Character> stack = new Stack<Character>();
    for(int i=0; i<str.length(); i++){
        char ch = str.charAt(i);

        if(Character.isLetterOrDigit(ch))
            result += ch;
        else if(ch == '(')
            stack.push(ch);
        else if(ch == ')'){
            while(!stack.isEmpty() && stack.peek()!='(')
                result += stack.pop();

            if(!stack.isEmpty() && stack.peek()!='(')
                return null;
            else
                stack.pop();
        }
        else {
            while(!stack.isEmpty() && priority(ch) <= priority(stack.peek()))
                result += stack.pop();
            stack.push(ch);
        }
    }
    while(!stack.isEmpty())
        result += stack.pop();

    return result;
}

public int postfixEvaluation(String str){
    if(str == null || str.length() == 0)
        return -1;
    Stack<Integer> stack = new Stack<>();
    for(int i=0; i<str.length(); i++){
        char ch = str.charAt(i);

        if(Character.isDigit(ch))
            stack.push(ch - '0');
        else {
            int val1 = stack.pop();
            int val2 = stack.pop();

            switch(ch){
                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 str = "[{(a+b)-(c*d)} + 2]";
        String str2 = "[{(a+b)-(c*d) + 2]";

        BalancedParenthesis obj = new BalancedParenthesis();
        System.out.println("Balanced or not: " + obj.checkIfBalanced(str));
        System.out.println("Balanced or not: " + obj.checkIfBalanced(str2));

        String str3 = "a+b*(c^d-e)^(f+g*h)-i";
        System.out.println(obj.infixToPostfix(str3));
        System.out.println(obj.postfixEvaluation("231*+9-"));
    }
}


import java.util.Stack;

public class AdvancedStack {

    Stack<Integer> elementStack = new Stack<>();
    Stack<Integer> minStack = new Stack<>();

    public void push(int data){
        elementStack.push(data);
        if(minStack.isEmpty() || minStack.peek() >= data)
            minStack.push(data);
    }

    public int pop(){
        if(elementStack.isEmpty())
            return -1;
        int item = elementStack.pop();
        int min = minStack.peek();
        if(min == item)
            minStack.pop();
        return item;
    }

    public int top(){
        return elementStack.peek();
    }

    public int minimum(){
        return minStack.peek();
    }

    public boolean isEmpty(){
        return elementStack.isEmpty();
    }

    public static void main(String[] args) {
        AdvancedStack advancedStack = new AdvancedStack();
        advancedStack.push(2);
        advancedStack.push(6);
        advancedStack.push(4);
        advancedStack.push(1);
        advancedStack.push(5);
        advancedStack.push(0);
```

```
        System.out.println("Element stack top: " + advancedStack.top());
        System.out.println("Current minimum: " + advancedStack.minimum());
        advancedStack.pop();
        advancedStack.pop();
        System.out.println("Element stack top: " + advancedStack.top());
        System.out.println("Current minimum: " + advancedStack.minimum());
        advancedStack.pop();
        System.out.println("Element stack top: " + advancedStack.top());
        System.out.println("Current minimum: " + advancedStack.minimum());
    }
}
```

```
import java.util.Stack;

public class StockSpan {

    public void calculateSpan(int price[], int span[]){
        Stack<Integer> stack = new Stack<>();
        span[0] = 1;
        stack.push(0);
        for(int i=1; i<price.length; i++){
            while (!stack.isEmpty() && price[stack.peek()] <= price[i])
                stack.pop();
            if(stack.isEmpty())
                span[i] = i+1;
            else
                span[i] = i-stack.peek();
            stack.push(i);
        }
    }

    public static void main(String[] args) {
        int price[] = {100, 80, 60, 70, 60, 75, 85};
        int span[] = new int[price.length];
        StockSpan ss = new StockSpan();
        ss.calculateSpan(price,span);
        for(int i=0;i<span.length;i++){
            System.out.print(span[i] + " ");
        }
        System.out.println();
    }
}
```

```
import java.util.Stack;

public class MaximumAreaHistogram {

    public int maxHistogram(int hist[]){
        Stack<Integer> stack = new Stack<>();
        int maxArea = 0;
        int area = 0;
        int i;
        for(i=0;i<hist.length;){
            if(stack.isEmpty() || hist[stack.peek()]<= hist[i])
                stack.push(i++);
            else{
                int top = stack.pop();
```

```
                if(stack.isEmpty())
                    area = hist[top]*i;
                else
                    area = hist[top] * (i-stack.peek()-1);
            }
            if(area > maxArea)
                maxArea = area;
        }
        while (!stack.isEmpty()){
            int top = stack.pop();
            if(stack.isEmpty())
                area = hist[top]*i;
            else
                area = hist[top] * (i-stack.peek()-1);
            if(area > maxArea)
                maxArea = area;
        }
        return maxArea;
    }

    public static void main(String[] args) {
        int hist[] = {2,1,2,3,1};
        MaximumAreaHistogram obj = new MaximumAreaHistogram();
        System.out.println("Max Area: " + obj.maxHistogram(hist));
    }
}
```