**STACK**

**Question 1:**

**Problem Statement( Valid string):**

Write a Java program to check whether a given string of brackets is *balanced*. A string is considered balanced if every opening bracket has a corresponding and correctly ordered closing bracket.

The valid brackets are:

* ()
* {}
* []

**Program:**

import java.util.Stack;

class Main{

public static boolean isValid(String s){

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

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

char ch=s.charAt(i);

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

stack.push(ch);

}else{

if(stack.isEmpty())return false;

char top=stack.pop();

if((ch==')'&&ch=='(')||(ch=='['&&ch==']')||(ch=='{'&&ch=='}')){

return false;

} } }

return stack.isEmpty();

} public static void main(String args[]){

String s="(({}{[[]]}))";

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

}

}

**Question 2:**

**Problem Statement:**

Write a Java program to evaluate a **postfix expression** (also known as Reverse Polish Notation). A postfix expression is a mathematical expression where each operator follows all of its operands.

You are given a string s consisting of **single-digit operands** (0–9) and **operators**: '+', '-', '\*', '/'. Your task is to evaluate the expression and return the result as an integer.

**Program:**

import java.util.Stack;

public class Main{

public static int isValid(String s){

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

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

char ch=s.charAt(i);

if(Character.isDigit(ch))

stack.push(ch-'0');

else{

int b=stack.pop();

int a=stack.pop();

int r=0;

switch(ch){

case '+':

r= a+b;

break;

case '-':

r=a-b;

break;

case '\*':

r= a\*b;

break;

case '/':

r= a/b;

break;

}stack.push(r);

}

}return stack.pop();

}

public static void main(String args[]){

String s="56+34-\*62/+";

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

}

}

**Question 3:**

**Problem Statement:**

Write a Java program to evaluate a **prefix expression** (also known as Polish Notation). In a prefix expression, **operators precede their operands**. For example, the expression + 2 3 evaluates to 5.

You are given a string s consisting of **single-digit operands** (0–9) and **operators**: '+', '-', '\*', '/',’^’. Your task is to evaluate the expression and return the result as an integer.

**Program:**

import java.util.Stack;

public class Main{

public static int Prefix(String s){

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

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

char ch=s.charAt(i);

if(Character.isDigit(ch))

stack.push(ch-'0');

else{

int a=stack.pop();

int b=stack.pop();

int r=0;

switch(ch){

case '+':

r=a+b;

break;

case '-':

r=a-b;

break;

case '\*':

r=a\*b;

break;

case '/':

r=a/b;

break;

case '^':

r=(int)Math.pow(a,b);

break;

}stack.push(r);

}

}return stack.pop();

}

public static void main(String args[]){

String s="-+\*823/^64+31";

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

}

}

**Question 4:**

**Problem Statement:**

Write a Java program to evaluate an **infix expression** containing:

* single-digit integers (0–9)
* operators: +, -, \*, /, ^
* parentheses: ( and )

The expression follows standard **arithmetic precedence** and **associativity rules**:

* ^ has the highest precedence and is **right-associative**
* \*, / have medium precedence and are **left-associative**
* +, - have the lowest precedence and are **left-associative**

**Program:**

import java.util.Stack;

public class Main {

public static int precedence(char ch) {

if (ch == '+' || ch == '-') return 1;

if (ch == '\*' || ch == '/') return 2;

if (ch == '^') return 3;

return 0;

}

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

switch (op) {

case '+': return a + b;

case '-': return a - b;

case '\*': return a \* b;

case '/': return a / b;

case '^': return (int) Math.pow(a, b);

}

return 0;

}

public static int infix(String s) {

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

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

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

char ch = s.charAt(i);

if (Character.isDigit(ch)) {

v.push(ch - '0');

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

ops.push(ch);

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

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

int b = v.pop();

int a = v.pop();

char op = ops.pop();

v.push(calc(a, b, op));

} ops.pop();

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

while (!ops.isEmpty() && ops.peek() != '(' &&

(precedence(ops.peek()) > precedence(ch) ||

(precedence(ops.peek()) == precedence(ch) && ch != '^'))) {

int b = v.pop();

int a = v.pop();

char op = ops.pop();

v.push(calc(a, b, op));

}ops.push(ch);

}

}

while (!ops.isEmpty()) {

int b = v.pop();

int a = v.pop();

char op = ops.pop();

v.push(calc(a, b, op));

}return v.pop();

}

public static void main(String[] args) {

String s = "(8\*2)+3-(6^4)/(3+1)";

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

}

}