

Mohammad Ali Jinnah University

Chartered by Government of Sindh - Recognized by HEC

**Assignment 1**

**Name:** Muhamad Fahad

**Id:** FA19-BSSE-0014

**Subject:** Data Structures and Algorithms Lab (CS 2511)

**Section:** AM

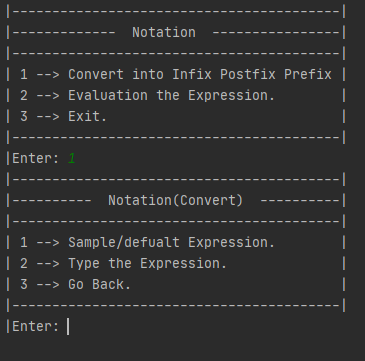
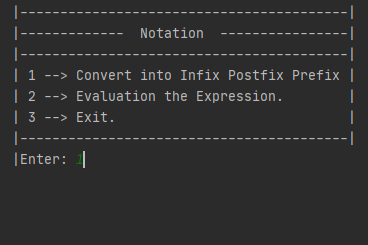
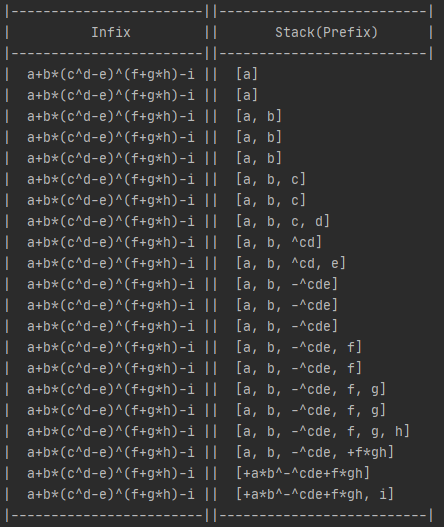
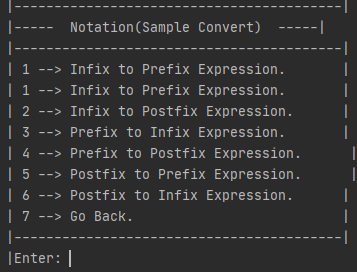
**Teacher:** MUHAMMAD MUBASHIR KHAN

**Date:** Sunday, December 20, 2020

# Task:

# **Present Polish notations (Prefix & Postfix) implementation using Stack.**

**Output:**

And other are Example are given in the code.

**Code:**

package com.company.Linkedlist;  
  
import java.io.IOException;  
import java.util.Scanner;  
import java.util.Stack;  
import java.util.Timer;  
import java.util.TimerTask;  
  
class Converter{  
 // PostFix to exp Infix  
 boolean isOperand(char charAt) {  
 return (charAt >= 'a' && charAt <= 'z') || (charAt >= 'A' && charAt <= 'Z');  
 }  
 int Prec(char ch) {  
 switch (ch) {  
 case '+':  
 case '-':  
 return 1;  
  
 case '\*':  
 case '/':  
 return 2;  
  
 case '^':  
 return 3;  
 }  
 return -1;  
 }  
  
 //Infix Operation  
 String InToPost(String exp){  
 String result = "";  
 char c;  
 Stack<Character> stack = new Stack<>();  
  
 for (int i = 0; i < exp.length(); ++i){  
 c = exp.charAt(i);  
  
 if (Character.*isLetterOrDigit*(c))  
 result += c;  
  
 else if (c == '(')  
 stack.push(c);  
  
 else if (c == ')'){  
 while (!stack.isEmpty() && stack.peek() != '(')  
 result += stack.pop();  
  
 stack.pop();  
 }  
 else {  
 while (!stack.isEmpty() && Prec(c) <= Prec(stack.peek()))  
 result += stack.pop();  
  
 stack.push(c);  
 }  
 System.*out*.println("| "+exp+" || "+result);  
 }  
  
 while (!stack.isEmpty()){  
 if(stack.peek() == '(')  
 return "Invalid Expression";  
 result += stack.pop();  
 }  
  
 return result;  
 }  
 String inToPre(String exp) {  
 Stack<Character> operators = new Stack<Character>();  
 Stack<String> operands = new Stack<String>();  
 String op1,op2;  
 char op, c;  
  
 for (int i = 0; i < exp.length(); i++) {  
 c = exp.charAt(i);  
 if (c == '(')  
 operators.push(c);  
  
 else if (c == ')') {  
 while (!operators.empty() && operators.peek() != '(') {  
 op1 = operands.pop();  
 op2 = operands.pop();  
 op = operators.pop();  
  
 operands.push(op + op2 + op1);  
 }  
 operators.pop();  
 }  
  
 else if (isOperand(c))  
 operands.push(c + "");  
 else {  
 while (!operators.empty() && Prec(c) <= Prec(operators.peek())) {  
 op1 = operands.pop();  
 op2 = operands.pop();  
 op = operators.pop();  
  
 operands.push(op + op2 + op1);  
 }  
  
 operators.push(c);  
 }  
 System.*out*.println("| "+exp+" || "+operands.toString());  
 }  
  
 while (!operators.empty()){  
 op1 = operands.pop();  
 op2 = operands.pop();  
 op = operators.pop();  
  
 operands.push(op + op2 + op1);  
 }  
 return operands.peek();  
 }  
  
 //PostFix Operation  
 String PostToIn(String exp){  
 Stack<String> s = new Stack<String>();  
 char c;  
 for(int i = 0; i < exp.length(); i++){  
 c = exp.charAt(i);  
 if (isOperand(c)){  
 s.push(c + "");  
 }  
 else{  
 String b = s.pop();  
 String a = s.pop();  
 s.push("(" + a + c + b + ")");  
 }  
  
 System.*out*.println("| "+exp+" || "+s.toString());  
 }  
  
 return s.peek();  
 }  
 String PostToPre(String exp) {  
 Stack<String> s = new Stack<>();  
 String op2,op1;  
 char c;  
  
 for (int i = 0; i < exp.length(); i++) {  
 c = exp.charAt(i);  
 if (!isOperand(c)) {  
 op1 = s.pop();  
 op2 = s.pop();  
  
 s.push(c + op2 + op1);  
 }  
 else  
 s.push(c+ "");  
  
 System.*out*.println("| "+exp+" || "+s.toString());  
 }  
  
 return s.toString();  
 }  
  
 //Prefix Operation  
 String PreToIn(String exp){  
 Stack<String> s = new Stack<>();  
 char c;  
 String op1 ,op2;  
 for(int i = exp.length()-1; i >= 0; i--){  
 c = exp.charAt(i);  
  
 if (!isOperand(c)) {  
 op1 = s.pop();  
 op2 = s.pop();  
  
 s.push(")" + op2 + c + op1 + "(");  
 }  
  
 else s.push(c + "");  
 System.*out*.println("| "+exp+" || "+(new StringBuilder(s.toString()).reverse())+"\b");  
 }  
  
 StringBuilder temp = (new StringBuilder(s.peek())).reverse();  
  
 return temp.toString();  
 }  
 String PreToPost(String exp){  
 Stack<String> s = new Stack<>();  
 char c;  
 String op1 ,op2;  
 for(int i = exp.length()-1; i >= 0; i--){  
 c = exp.charAt(i);  
  
 if (!isOperand(c)) {  
 op1 = s.pop();  
 op2 = s.pop();  
  
 s.push( op1 +op2+ c );  
 }  
  
 else s.push(c + "");  
 System.*out*.println("| "+exp+" || "+s.toString());  
 }  
  
 return s.peek();  
 }  
  
 int calculatePost(String exp){  
 Stack<Integer> stack = new Stack<>();  
 char c;  
 int oper1, oper2;  
  
 for (int i = 0; i < exp.length(); i++) {  
 c = exp.charAt(i);  
  
 if(Character.*isDigit*(c))  
 stack.push(Integer.*parseInt*(String.*valueOf*(c)));  
 else {  
 oper1 = stack.pop();  
 oper2 = stack.pop();  
  
 stack.push((c == '+'?(oper2+oper1):(c == '-'?(oper2-oper1):(c == '\*'?(oper2\*oper1):(c == '/'?(oper2/oper1):0)))));  
 }  
  
 }  
  
 return stack.pop();  
 }  
 int calculatePre(String exp){  
 Stack<Integer> stack = new Stack<>();  
 char c;  
 int oper1, oper2;  
  
 for (int i = exp.length()-1; i >= 0; i--) {  
 c = exp.charAt(i);  
  
 if(Character.*isDigit*(c))  
 stack.push(Integer.*parseInt*(String.*valueOf*(c)));  
 else {  
 oper1 = stack.pop();  
 oper2 = stack.pop();  
  
 stack.push((c == '+'?(oper1+oper2):(c == '-'?(oper1-oper2):(c == '\*'?(oper1\*oper2):(c == '/'?(oper1/oper2):0)))));  
 }  
  
 }  
  
 return stack.pop();  
 }  
}  
  
public class Question2 {  
 public static void main(String[] args) throws IOException, InterruptedException {  
 Converter Menu = new Converter();  
 Scanner scan = new Scanner(System.*in*);  
 String exp;  
 int k1,k2;  
 boolean condition = true;  
 while (condition){  
 System.*out*.println("|-----------------------------------------|");  
 System.*out*.println("|------------- Notation ----------------|");  
 System.*out*.println("|-----------------------------------------|");  
 System.*out*.println("| 1 --> Convert into Infix Postfix Prefix |");  
 System.*out*.println("| 2 --> Evaluation the Expression. |");  
 System.*out*.println("| 3 --> Exit. |");  
 System.*out*.println("|-----------------------------------------|");  
 System.*out*.print("|Enter: ");  
 k1 = scan.nextInt();  
 switch (k1){  
 case 3:  
 System.*exit*(0);  
 case 2:  
 while (condition) {  
 System.*out*.println("|-----------------------------------------|");  
 System.*out*.println("|--------- Notation(Evaluation) --------|");  
 System.*out*.println("|-----------------------------------------|");  
 System.*out*.println("| 1 --> Sample/defualt Expression. |");  
 System.*out*.println("| 2 --> Type the Expression. |");  
 System.*out*.println("| 3 --> Go Back. |");  
 System.*out*.println("|-----------------------------------------|");  
 System.*out*.print("|Enter: ");  
 k2 = scan.nextInt();  
 switch (k2) {  
 case 3:  
 condition = false;  
 break;  
 case 1:  
 while (condition) {  
 System.*out*.println("|-----------------------------------------|");  
 System.*out*.println("|----- Notation(Sample Evaluation) -----|");  
 System.*out*.println("|-----------------------------------------|");  
 System.*out*.println("| 1 --> Sample Prefix Expression. |");  
 System.*out*.println("| 2 --> Sample Prefix Expression. |");  
 System.*out*.println("| 3 --> Go Back. |");  
 System.*out*.println("|-----------------------------------------|");  
 System.*out*.print("|Enter: ");  
 switch (scan.nextInt()){  
 case 3:  
 condition = !condition;  
 break;  
 case 2:  
 exp="+9\*26";  
 System.*out*.println("postfix evaluation: "+Menu.calculatePre(exp));  
 break;  
 case 1:  
 exp="291\*+8/";  
 System.*out*.println("postfix evaluation: "+Menu.calculatePost(exp));  
 break;  
 default:  
 System.*out*.println("Invalid Input! ");  
 }  
 }  
 condition = true;  
 break;  
 case 2:  
 while (condition) {  
 System.*out*.println("|-----------------------------------------|");  
 System.*out*.println("|----- Notation(Type to Evaluation) ----|");  
 System.*out*.println("|-----------------------------------------|");  
 System.*out*.println("| 1 --> Type Prefix Expression. |");  
 System.*out*.println("| 2 --> Type Prefix Expression. |");  
 System.*out*.println("| 3 --> Go Back. |");  
 System.*out*.println("|-----------------------------------------|");  
 System.*out*.print("|Enter: ");  
 switch (scan.nextInt()){  
 case 3:  
 condition = !condition;  
 break;  
 case 2:  
 exp = scan.nextLine();  
 System.*out*.println("postfix evaluation: "+Menu.calculatePre(exp));  
 break;  
 case 1:  
 exp = scan.nextLine();  
 System.*out*.println("postfix evaluation: "+Menu.calculatePost(exp));  
 break;  
 default:  
 System.*out*.println("Invalid Input! ");  
 }  
 }  
 condition = true;  
 break;  
 default:  
 System.*out*.println("Invalid Input! ");  
 }  
 }  
 condition = true;  
 break;  
 case 1:  
 while (condition){  
 System.*out*.println("|-----------------------------------------|");  
 System.*out*.println("|---------- Notation(Convert) ----------|");  
 System.*out*.println("|-----------------------------------------|");  
 System.*out*.println("| 1 --> Sample/defualt Expression. |");  
 System.*out*.println("| 2 --> Type the Expression. |");  
 System.*out*.println("| 3 --> Go Back. |");  
 System.*out*.println("|-----------------------------------------|");  
 System.*out*.print("|Enter: ");  
 k2 = scan.nextInt();  
 switch (k2) {  
 case 3:  
 condition = false;  
 break;  
 case 1:  
 while (condition) {  
 System.*out*.println("|-----------------------------------------|");  
 System.*out*.println("|----- Notation(Sample Convert) -----|");  
 System.*out*.println("|-----------------------------------------|");  
 System.*out*.println("| 1 --> Infix to Prefix Expression. |"); System.*out*.println("| 1 --> Infix to Prefix Expression. |");  
 System.*out*.println("| 2 --> Infix to Postfix Expression. |");  
 System.*out*.println("| 3 --> Prefix to Infix Expression. |");  
 System.*out*.println("| 4 --> Prefix to Postfix Expression. |");  
 System.*out*.println("| 5 --> Postfix to Prefix Expression. |");  
 System.*out*.println("| 6 --> Postfix to Infix Expression. |");  
 System.*out*.println("| 7 --> Go Back. |");  
 System.*out*.println("|-----------------------------------------|");  
 System.*out*.print("|Enter: ");  
 switch (scan.nextInt()){  
 case 7:  
 condition = !condition;  
 break;  
 case 6:  
 // part 1B in which the postfix Convert in to infix(parameterized)  
 exp = "abcd^e-fgh\*+^\*+i-";  
 System.*out*.println("|------------------------||--------------------------|");  
 System.*out*.println("| PostFix || Stack(Infix) |");  
 System.*out*.println("|------------------------||--------------------------|");  
 exp = Menu.PostToIn(exp);  
 System.*out*.println("|------------------------||--------------------------|");  
 System.*out*.println("\nFinal output: "+exp);  
 break;  
 case 5:  
 // part 2B in which the postfix Convert in to prefix  
 exp = "abcd^e-fgh\*+^\*+i-";  
 System.*out*.println("|------------------------||--------------------------|");  
 System.*out*.println("| PostFix || Stack(Prefix) |");  
 System.*out*.println("|------------------------||--------------------------|");  
 exp = Menu.PostToPre(exp);  
 System.*out*.println("|------------------------||--------------------------|");  
 System.*out*.println("\nFinal output: "+exp);  
 break;  
 case 4:  
 // part 2C in which the prefix Convert in to postfix  
 exp = "-+a\*b^-^cde+f\*ghi";  
 System.*out*.println("|------------------------||--------------------------|");  
 System.*out*.println("| PreFix || Stack(Postfix) |");  
 System.*out*.println("|------------------------||--------------------------|");  
 exp = Menu.PreToPost(exp);  
 System.*out*.println("|------------------------||--------------------------|");  
 System.*out*.println("\nFinal output: "+exp);  
 break;  
 case 3:  
// part 1C in which the prefix Convert in to Infix  
 exp = "-+a\*b^-^cde+f\*ghi";  
 System.*out*.println("|------------------------||--------------------------|");  
 System.*out*.println("| Prefix || Stack(Infix) |");  
 System.*out*.println("|------------------------||--------------------------|");  
 exp = Menu.PreToIn(exp);  
 System.*out*.println("|------------------------||--------------------------|");  
 System.*out*.println("\nFinal output: "+exp);  
 break;  
 case 2:  
// part 2A in which the infix(parameterized) Convert in to Prefix  
 exp = "a+b\*(c^d-e)^(f+g\*h)-i";  
 System.*out*.println("|------------------------||--------------------------|");  
 System.*out*.println("| Infix || Stack(Prefix) |");  
 System.*out*.println("|------------------------||--------------------------|");  
 exp = Menu.inToPre(exp);  
 System.*out*.println("|------------------------||--------------------------|");  
 System.*out*.println("\nFinal output: "+exp);  
 break;  
 case 1:  
// part 1A in which the infix(parameterized) Convert in to postfix  
 exp = "a+b\*(c^d-e)^(f+g\*h)-i";  
 System.*out*.println("|------------------------||--------------------------|");  
 System.*out*.println("| Infix || Stack(Postfix) |");  
 System.*out*.println("|------------------------||--------------------------|");  
 exp = Menu.InToPost(exp);  
 System.*out*.println("|------------------------||---------------------------|");  
 System.*out*.println("\nFinal output: "+exp);  
 break;  
 default:  
 System.*out*.println("Invalid Input! ");  
 }  
 }  
 condition = true;  
 break;  
 case 2:  
 while (condition) {  
 System.*out*.println("|-----------------------------------------|");  
 System.*out*.println("|----- Notation(Type Convert) -----|");  
 System.*out*.println("|-----------------------------------------|");  
 System.*out*.println("| 1 --> Infix to Prefix Expression. |"); System.*out*.println("| 1 --> Infix to Prefix Expression. |");  
 System.*out*.println("| 2 --> Infix to Postfix Expression. |");  
 System.*out*.println("| 3 --> Prefix to Infix Expression. |");  
 System.*out*.println("| 4 --> Prefix to Postfix Expression. |");  
 System.*out*.println("| 5 --> Postfix to Prefix Expression. |");  
 System.*out*.println("| 6 --> Postfix to Infix Expression. |");  
 System.*out*.println("| 7 --> Go Back. |");  
 System.*out*.println("|-----------------------------------------|");  
 System.*out*.print("|Enter: ");  
 switch (scan.nextInt()){  
 case 7:  
 condition = !condition;  
 break;  
 case 6:  
 // part 1B in which the postfix Convert in to infix(parameterized)  
 exp = scan.nextLine();  
 System.*out*.println("|------------------------||--------------------------|");  
 System.*out*.println("| PostFix || Stack(Infix) |");  
 System.*out*.println("|------------------------||--------------------------|");  
 exp = Menu.PostToIn(exp);  
 System.*out*.println("|------------------------||--------------------------|");  
 System.*out*.println("\nFinal output: "+exp);  
 break;  
 case 5:  
 // part 2B in which the postfix Convert in to prefix  
 exp = scan.nextLine();  
 System.*out*.println("|------------------------||--------------------------|");  
 System.*out*.println("| PostFix || Stack(Prefix) |");  
 System.*out*.println("|------------------------||--------------------------|");  
 exp = Menu.PostToPre(exp);  
 System.*out*.println("|------------------------||--------------------------|");  
 System.*out*.println("\nFinal output: "+exp);  
 break;  
 case 4:  
 // part 2C in which the prefix Convert in to postfix  
 exp = scan.nextLine();  
 System.*out*.println("|------------------------||--------------------------|");  
 System.*out*.println("| PreFix || Stack(Postfix) |");  
 System.*out*.println("|------------------------||--------------------------|");  
 exp = Menu.PreToPost(exp);  
 System.*out*.println("|------------------------||--------------------------|");  
 System.*out*.println("\nFinal output: "+exp);  
 break;  
 case 3:  
// part 1C in which the prefix Convert in to Infix  
 exp = scan.nextLine();  
 System.*out*.println("|------------------------||--------------------------|");  
 System.*out*.println("| Prefix || Stack(Infix) |");  
 System.*out*.println("|------------------------||--------------------------|");  
 exp = Menu.PreToIn(exp);  
 System.*out*.println("|------------------------||--------------------------|");  
 System.*out*.println("\nFinal output: "+exp);  
 break;  
 case 2:  
// part 2A in which the infix(parameterized) Convert in to Prefix  
 exp = scan.nextLine();  
 System.*out*.println("|------------------------||--------------------------|");  
 System.*out*.println("| Infix || Stack(Prefix) |");  
 System.*out*.println("|------------------------||--------------------------|");  
 exp = Menu.inToPre(exp);  
 System.*out*.println("|------------------------||--------------------------|");  
 System.*out*.println("\nFinal output: "+exp);  
 break;  
 case 1:  
// part 1A in which the infix(parameterized) Convert in to postfix  
 exp = scan.nextLine();  
 System.*out*.println("|------------------------||--------------------------|");  
 System.*out*.println("| Infix || Stack(Postfix) |");  
 System.*out*.println("|------------------------||--------------------------|");  
 exp = Menu.InToPost(exp);  
 System.*out*.println("|------------------------||---------------------------|");  
 System.*out*.println("\nFinal output: "+exp);  
 break;  
 default:  
 System.*out*.println("Invalid Input! ");  
 }  
 }  
 condition = true;  
 break;  
 default:  
 System.*out*.println("Invalid Input! ");  
 }  
 }  
 condition = true;  
 break;  
  
 }  
  
 }  
 }  
}