**Program Sixteen:**

//summary: This program takes in infix notation and changes it into postfix notation

//name: Jenna Wolf

//class: Fundamentals of Programming, CS155 - 01

//instructor: Dr. Art Kazmierczak

//date: 11/29/2023

import java.util.Scanner; //allows for inputs to be made

import java.util.ArrayList; //allows the arraylist class to be used

public class Main

{

public static void main(String[] args)

{

Scanner input = new Scanner(System.in); //names the input

String expression; //holds the expression data

//takes in an expression from the user

System.out.println("Please enter an expression");

expression = input.nextLine();

//outputs the original expression and the postfix expression (calls the postfix function)

System.out.println(expression + " in postfix notation is " + infixToPostfix(expression));

}

public static String infixToPostfix(String expression)

{

String postfix = ""; //holds the postfix data and sets it to blank

char variable; //holds the variable data

ArrayList<Character> operators = new ArrayList<>(); //holds the operators data

//loops until expression length is hit

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

{

variable = expression.charAt(i); //sets varaible to the current character

//if variable is a digit, add to postfix

if(Character.isDigit(variable))

postfix += variable + " ";

//if varaible is +, -, \*, /, adds to operators and adds to postfix based on precendence

else if(variable == '+' || variable == '-' || variable == '\*' || variable == '/')

{

while(!operators.isEmpty() && highPrecedence(operators.get(operators.size() - 1), variable))

postfix += operators.remove(operators.size() - 1) + " ";

operators.add(variable);

}

//if variable is (, adds it to operators

else if(variable == '(')

operators.add(variable);

//if varaible is ), adds all operators inside parentheses

else if(variable == ')')

{

while(!operators.isEmpty() && (operators.get(operators.size() - 1)) != '(')

{

postfix += operators.remove(operators.size() - 1) + " ";

}

if(!operators.isEmpty())

operators.remove(operators.size() - 1);

}

}

//adds the rest of the operators

while(!operators.isEmpty())

postfix += operators.remove(operators.size() - 1) + " ";

return postfix; //returns the postfix string

}

private static boolean highPrecedence(char char1, char char2)

{

int op1 = precedence(char1); //sets op1 to return value from precednece for char1

int op2 = precedence(char2); //sets op2 to return value from precednece for char2

return op1 > op2; //returns true or flase based on results

}

private static int precedence(char op)

{

if(op == '+' || op == '-') //returns one if + or -

return 1;

else if (op == '\*' || op == '/') //returns two if + or -

return 2;

else

return 0; //returns 0 if no varaible to compare to

}

}

**Output:**

