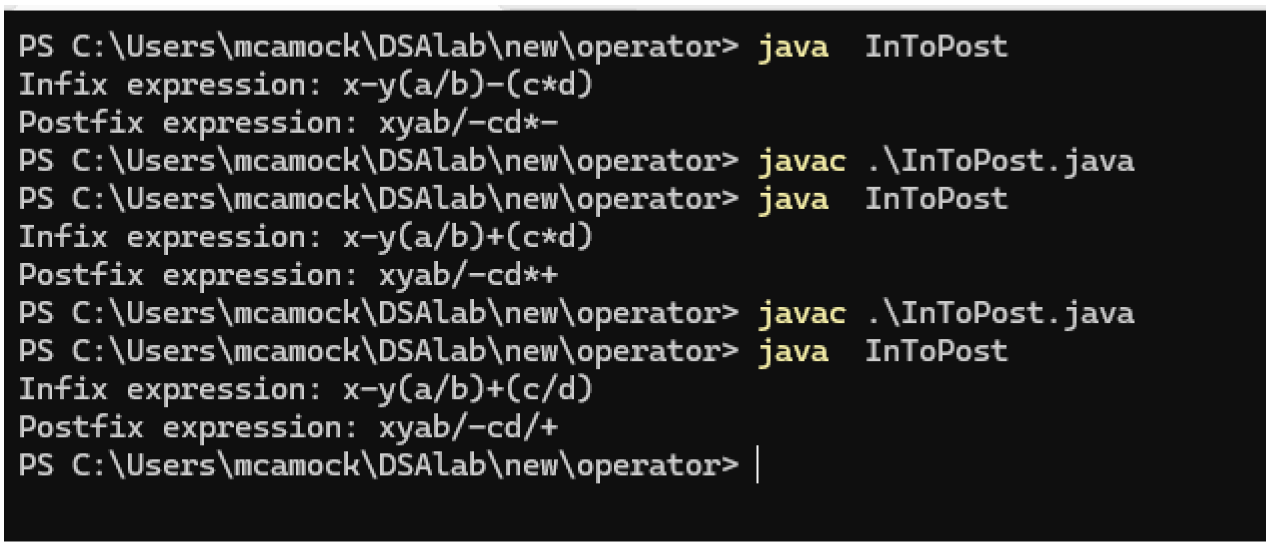
Program No:	**19**
Roll No :	**1525**
Title of Program :	Infix postfix
Objective :	Queue

SOURCE CODE:

```
import java.util.*;
public class InToPost
{
    //Operator check
    static boolean isOperator(char c)
    {
        return c=='+' || c=='-' || c=='*' || c=='/';
    }

    //Precedence
    static int precedence(char operator)
    {
        switch(operator)
        {
            case '+':
            case '-':
                    return 1;
            case '*':
            case '/':
                    return 2;
            default:
                    return -1;

        }

    }//end of precendence


    //Convert infix to postfix
    public static String infixToPostfix(String infix)
    {
        char[] stack=new char[infix.length()];
        int tos=-1;
        StringBuilder postfix= new StringBuilder();

        for (int i=0;i<infix.length();i++)
        {
            char ch=infix.charAt(i);
            if(Character.isLetterOrDigit(ch))
            {
                //Operand - Append to postfix - step 2
                postfix.append(ch);
            }
            else if(ch=='(')
            {
                //Open paranthesis - Push - step 3
                tos++;
                stack[tos]=ch;
            }

            else if (ch==')')
            {
                //close parenthesis - pop till ')' - step 5
                while(stack[tos]!='(')
                {
                    postfix.append(stack[tos]);
                    tos--;
                }//end of while
                tos--; //pop the '('
            }
            else if(isOperator(ch))
            {
                //operator - pop till step 3b
                while(tos>=0 && precedence(ch) <= precedence(stack[tos]))
                {
                    postfix.append(stack[tos]);
                    tos--;
                }//end of while
                //push current ch
                tos++;
                stack[tos]=ch;
            }
        }//end of for loop

        //Pop Remaining characters from stack
        while(tos>=0)
        {
            postfix.append(stack[tos]);
            tos--;
        }
        return postfix.toString();
    }//end of infixToPostfix

    //Main
    public static void main(String[] args)
    {
        String infix="x-y(a/b)-(c*d)";

        String result=infixToPostfix(infix);
        System.out.println("Infix expression: "+infix);
        System.out.println("Postfix expression: "+result);
    }
}
```

OUTPUT:



```
PS C:\Users\mcamock\DSAlab\new\operator> java  InToPost
Infix expression: x-y(a/b)-(c*d)
Postfix expression: xyab/-cd*-
PS C:\Users\mcamock\DSAlab\new\operator> javac .\InToPost.java
PS C:\Users\mcamock\DSAlab\new\operator> java  InToPost
Infix expression: x-y(a/b)+(c*d)
Postfix expression: xyab/-cd*+
PS C:\Users\mcamock\DSAlab\new\operator> javac .\InToPost.java
PS C:\Users\mcamock\DSAlab\new\operator> java  InToPost
Infix expression: x-y(a/b)+(c/d)
Postfix expression: xyab/-cd/+
PS C:\Users\mcamock\DSAlab\new\operator>
```