**Name : Tanishq Thuse**

**Branch : SY-CS(AI)**

**Div : B**

**Roll No. : 60**

**Subject : ADS Assignment-6**

**Title : Infix to Postfix conversion using Stack**

**Q1) Infix to PostFix**

**Code :**

import java.util.\*;

public class infixApp2 {

static int prec(char c) {

if (c == '^')

return 3;

else if (c == '/' || c == '\*')

return 2;

else if (c == '+' || c == '-')

return 1;

else

return -1;

}

static char associativity(char c) {

if (c == '^')

return 'R';

return 'L';

}

static void infixToPostfix(String s) {

StringBuilder result = new StringBuilder();

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

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

char c = s.charAt(i);

// If the character is an operand, add it to output

if ((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z') || (c >= '0' && c <= '9')) {

result.append(c);

}

// If the character is '(', push it to stack

else if (c == '(') {

stack.push(c);

}

// If the character is ')', pop and output from the stack

// until an '(' is encountered

else if (c == ')') {

while (!stack.isEmpty() && stack.peek() != '(') {

result.append(stack.pop());

}

stack.pop();

}

// An operator is encountered

else {

while (!stack.isEmpty() && (prec(s.charAt(i)) < prec(stack.peek()) ||

(prec(s.charAt(i)) == prec(stack.peek()) && associativity(s.charAt(i)) == 'L'))) {

result.append(stack.pop());

}

stack.push(c);

}

}

// Pop all the operators from the stack

while (!stack.isEmpty()) {

result.append(stack.pop());

}

System.out.println(result);

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the infix expression: ");

String exp = sc.nextLine();

System.out.println("Postfix expression: ");

infixToPostfix(exp);

}

}

**Output :**

