**12. Write a C program to implement the application of Stack (Notations)**

#include <stdio.h>

#include <ctype.h>

#include <string.h>

#include <stdlib.h>

#define MAX 100

char stack[MAX];

int top = -1;

void push(char x) {

stack[++top] = x;

}

char pop() {

return stack[top--];

}

int precedence(char x) {

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

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

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

return 0;

}

void infixToPostfix(char infix[], char postfix[]) {

int i, k = 0;

char x;

for (i = 0; infix[i]; i++) {

if (isalnum(infix[i])) {

postfix[k++] = infix[i];

} else if (infix[i] == '(') {

push(infix[i]);

} else if (infix[i] == ')') {

while ((x = pop()) != '(')

postfix[k++] = x;

} else {

while (top != -1 && precedence(stack[top]) >= precedence(infix[i]))

postfix[k++] = pop();

push(infix[i]);

}

}

while (top != -1)

postfix[k++] = pop();

postfix[k] = '\0';

}

int evalPostfix(char exp[]) {

int st[MAX], t = -1, i, op1, op2;

for (i = 0; exp[i]; i++) {

if (isdigit(exp[i])) {

st[++t] = exp[i] - '0';

} else {

op2 = st[t--];

op1 = st[t--];

switch (exp[i]) {

case '+': st[++t] = op1 + op2; break;

case '-': st[++t] = op1 - op2; break;

case '\*': st[++t] = op1 \* op2; break;

case '/': st[++t] = op1 / op2; break;

}

}

}

return st[t];

}

int main() {

char infix[MAX], postfix[MAX];

printf("Enter infix expression: ");

scanf("%s", infix);

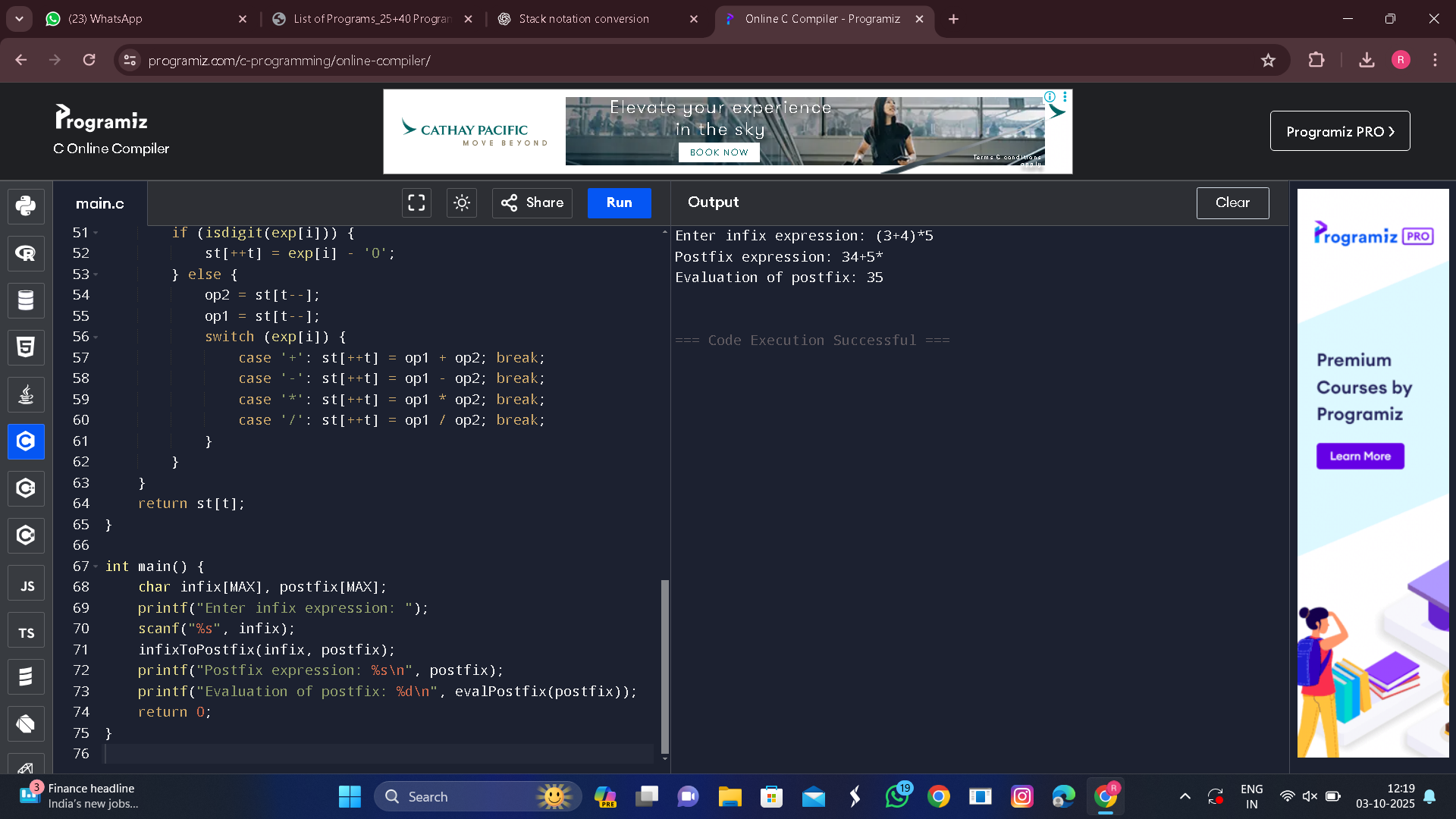
infixToPostfix(infix, postfix);

printf("Postfix expression: %s\n", postfix);

printf("Evaluation of postfix: %d\n", evalPostfix(postfix));

return 0;

}



**OUTPUT**