VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF ELECTRICAL AND ELECTRONICS ENGINEERING

COMPUTER SYSTEMS AND PROGRAMMING (EE2415)
CLASS TT01 - SEMESTER 222

PROJECT C

Student: Luong Hoang Phuc 2010525

HO CHI MINH CITY, MAY 2023

Write the program to make Reverse Polish Notation calculator like the following:

Infix	Reverse Polish notation
A + B × C	A B C × +
A × B + C	A B × C +
A × B + C × D	A B × C D × +
(A + B) / (C – D)	A B + C D – /
A × B / C	A B × C /
((A + B) × C + D)/(E + F + G)	A B + C × D + E F + G + /

Use functions and a dynamic stack to solve that with all checks.

Attention: this project's user interface is so important, let's organize it as well as possible for getting points.

```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>     //isdigit(int c): Checks if a character is a
decimal digit.

#define MAX_LENGTH 100  //used to specify the maximum length of character
arrays infix and postfix

/*The precedence function takes a character c as input and returns an
integer representing the precedence of the operator.
The higher the precedence value, the higher the precedence of the
operator.
The function checks the value of c and assigns a precedence value
accordingly. */
int precedence(char c) {
    if (c == '^')
        return 3;
    else if (c == '/' || c == '*')
        return 2;
    else if (c == '+' || c == '-')
        return 1;
    else
        return -1;
}

void infixToPostfix(char* infix, char* postfix) {
    char char_stack[MAX_LENGTH];
    int stack_top = -1, i = 0, j = 0;
    while (infix[i] != '\0') {
        char c = infix[i];
        if (isdigit(c) || isalpha(c) || c == '.') {
            while (isdigit(infix[i]) || isalpha(infix[i]) || infix[i] ==
```

```
'.') {
                postfix[j++] = infix[i++];
            }
            postfix[j++] = ' ';
            i--;
        }
        else if (c == ' ') {
            i++;
            continue;
        }
        else if (c == '(') {
            char_stack[++stack_top] = '(';
        }
        else if (c == ')') {
            while (char_stack[stack_top] != '(') {
                postfix[j++] = char_stack[stack_top--];
                postfix[j++] = ' ';
            }
            stack_top--;
        }
        else {
            while (stack_top >= 0 && precedence(c) <=
precedence(char_stack[stack_top])) {
                if (c == '^' && char_stack[stack_top] == '^')
                    break;
                else {
                    postfix[j++] = char_stack[stack_top--];
                    postfix[j++] = ' ';
                }
            }
            char_stack[++stack_top] = c;
        }
        i++;
    }
    while (stack_top >= 0) {
        postfix[j++] = char_stack[stack_top--];
        postfix[j++] = ' ';
    }
    postfix[j] = '\0';
}

int main() {
    char infix[MAX_LENGTH], postfix[MAX_LENGTH];
    printf("Enter infix expression: ");
    fgets(infix, MAX_LENGTH, stdin);
    infix[strlen(infix)-1] = '\0';
    infixToPostfix(infix, postfix);
    printf("Postfix expression: %s\n", postfix);
    return 0;
}
/*
The infixToPostfix function converts an infix expression to postfix
notation.
It takes two character array pointers as arguments: infix, representing
the input infix expression, and postfix,
which will store the resulting postfix expression.

Inside the infixToPostfix function, a character stack char_stack is
declared to
store operators during the conversion process.
```

The variable stack_top keeps track of the top element of the stack.

The function uses a while loop to iterate through each character in the infix expression
until it reaches the null termination character (\0).

The code checks the current character c in the infix expression.
If c is a digit, letter, or period (decimal point), it is appended to the postfix string until a non-digit, non-letter, or
non-period characters are encountered. This effectively extracts operands from the infix expression and adds them to the postfix expression.

If c is a space character, the loop continues to the next character without performing any further actions.

If c is an opening parenthesis '(', it is pushed onto the char_stack using the stack_top variable.

If c is a closing parenthesis ')', the code pops operators from the char_stack and appends them to the postfix string until
an opening parenthesis is encountered. The opening parenthesis is then discarded.

If c is an operator (e.g., '+', '-', '*', '/', '^'), the code checks the precedence of the operator against the operators in the char_stack.
It pops operators from the stack and appends them to the postfix string as long as the operator on the stack has higher
precedence or if the current operator is '^' (exponentiation) and the top of the stack
is also '^'. Finally, the current operator c is pushed onto the char_stack.

After processing all the characters in the infix expression, there may be remaining operators in the char_stack.
The code pops these operators from the stack and appends them to the postfix string.

Finally, the null termination character '\0' is added to the postfix string to indicate its end.

In the main function, the code declares character arrays infix and postfix to store the input
infix expression and the resulting postfix expression, respectively.

The user is prompted to enter an infix expression using the printf function.

The fgets function is used to read the infix expression from the user, limiting the input to MAX_LENGTH - 1 characters.

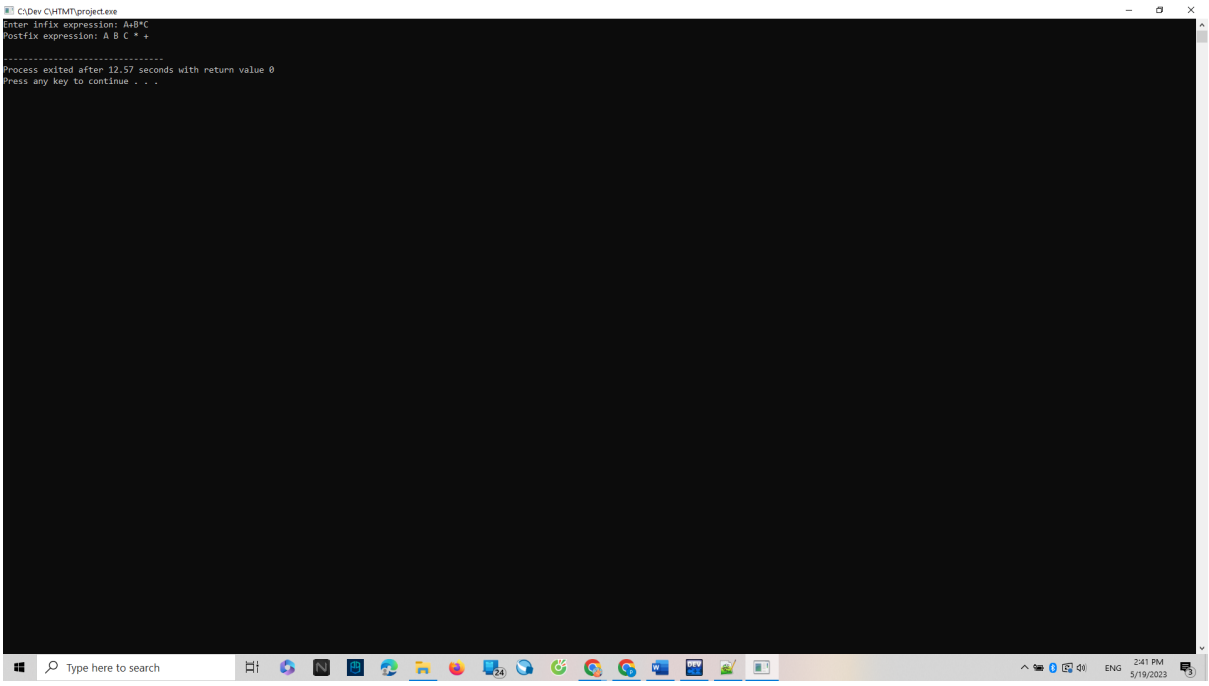
The newline character at the end of the input string is replaced with the null termination character ('\0')
using infix[strlen(infix) - 1] = '\0'. This is done to remove the newline character from the string.

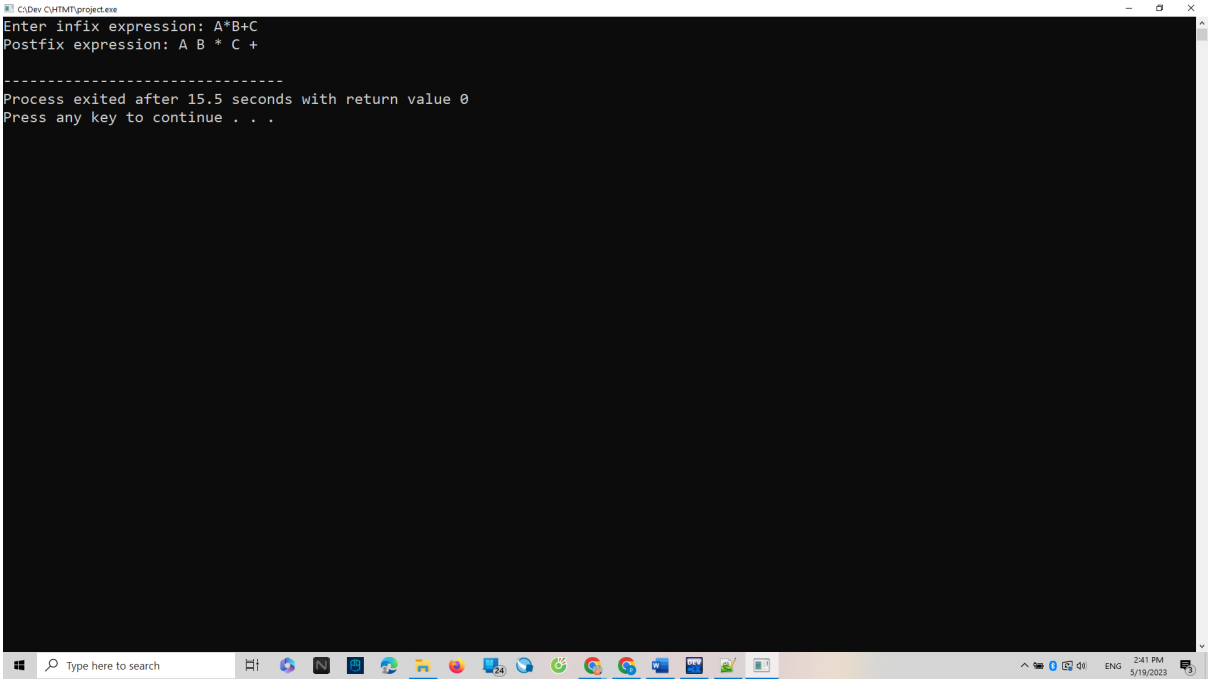
The infixToPostfix function is called with the infix and postfix arrays as arguments to convert the infix expression to postfix.

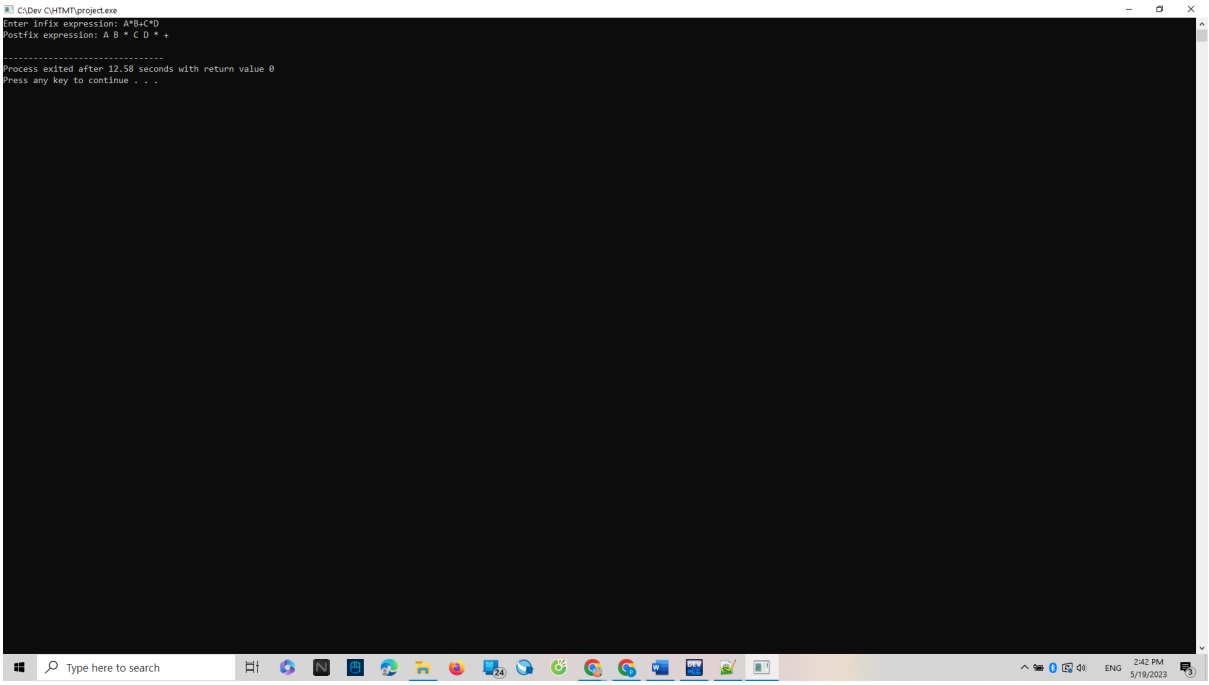
The resulting postfix expression is printed using printf with the format specifier %s.

```
The main function ends, and the program terminates.
*/
```

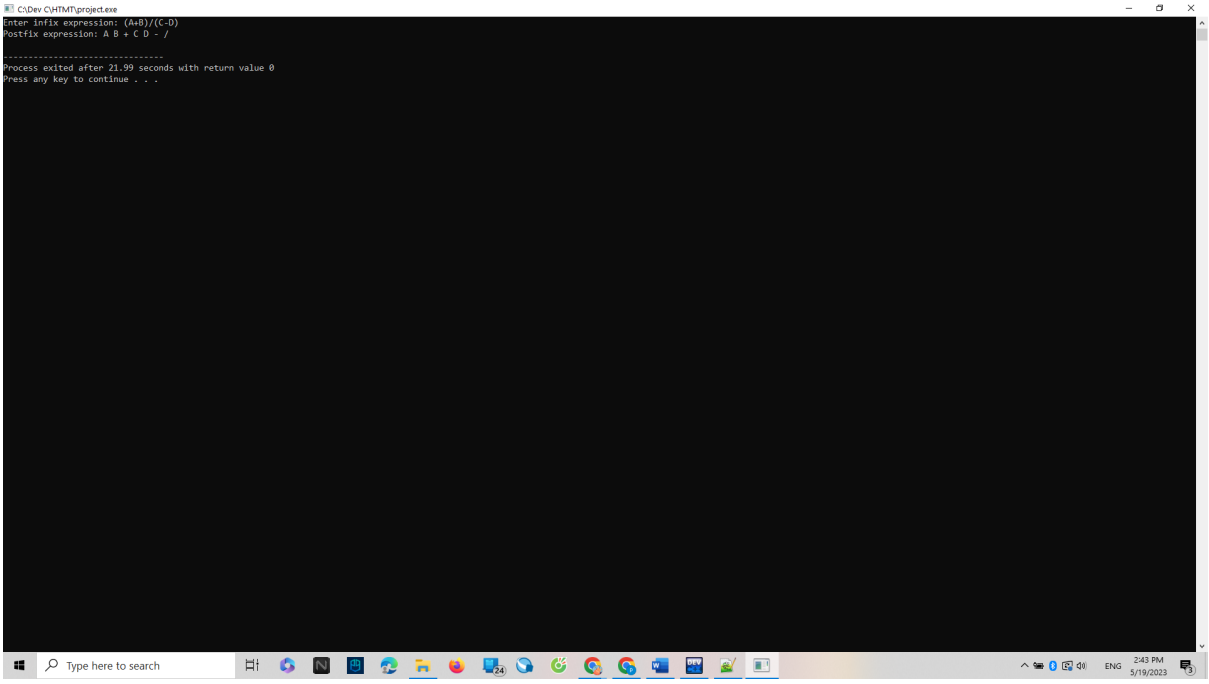
SIMULATION



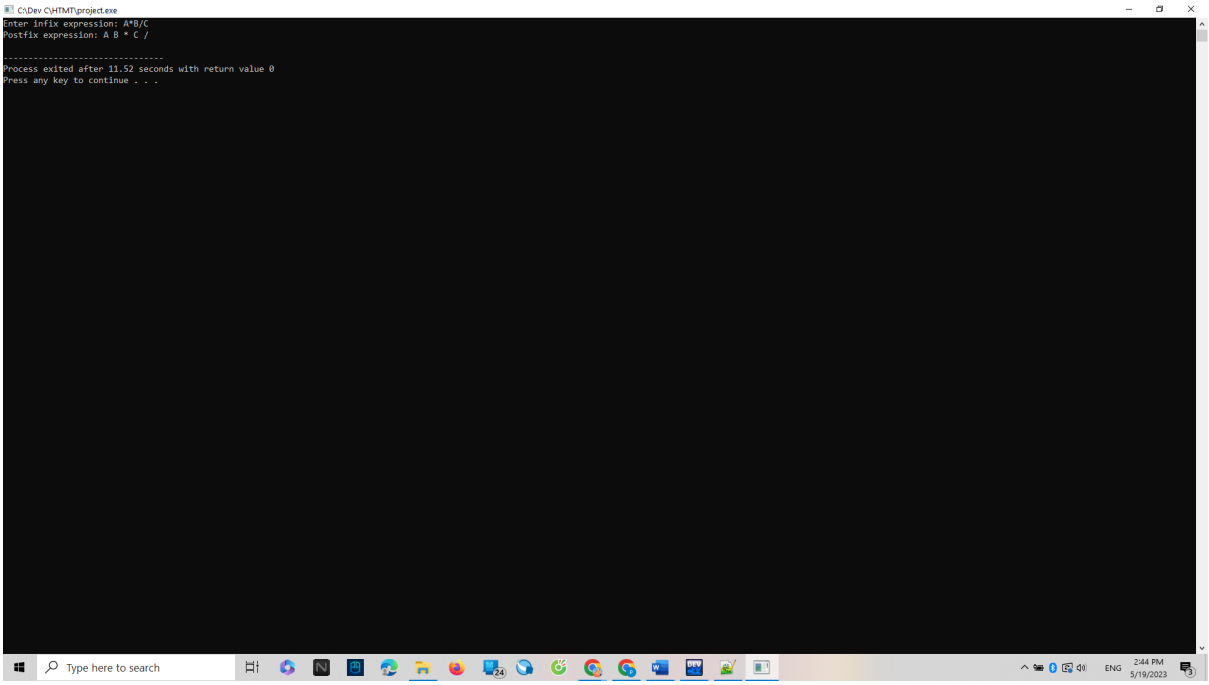



C:\Dev C\HTMT\project.exe
Enter infix expression: A*B+C*D
Postfix expression: A B * C D * +

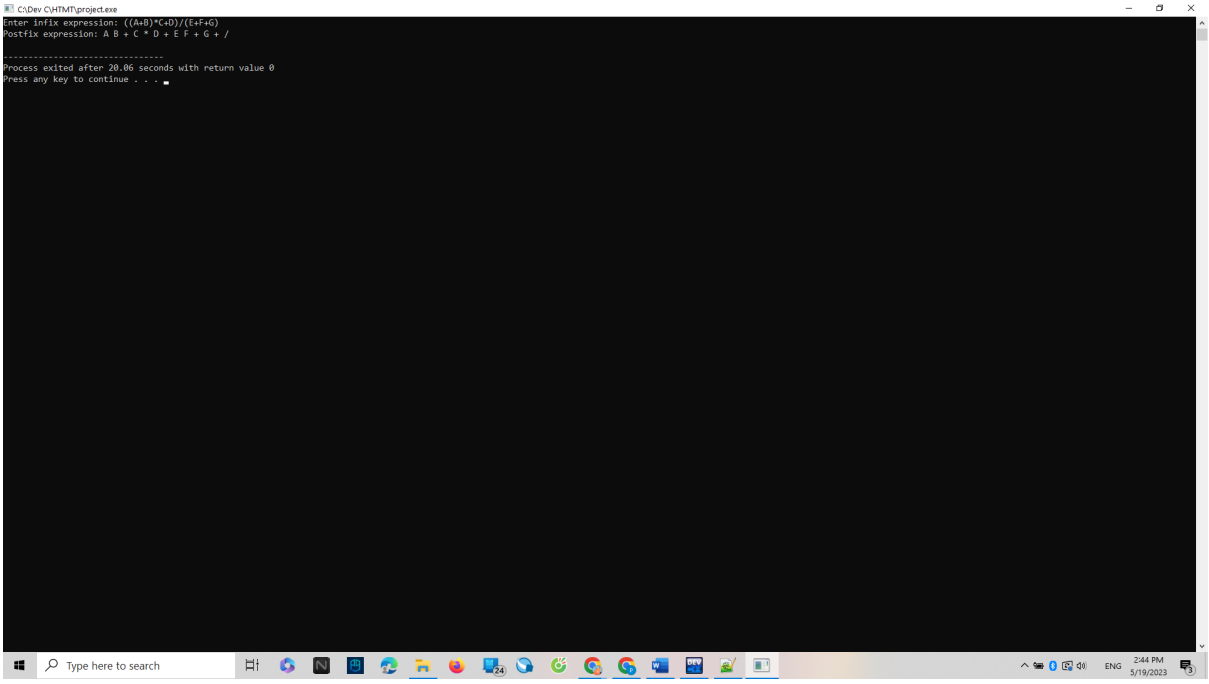
Process exited after 12.58 seconds with return value 0
Press any key to continue . . .
Type here to search
ENG
2:42 PM
5/19/2023


C:\Dev C\HTMT\project.exe
Enter infix expression: (A+B)/(C-D)
Postfix expression: A B + C D - /

Process exited after 21.99 seconds with return value 0
Press any key to continue . . .
Type here to search
ENG
2:43 PM
5/19/2023


C:\Dev C\HTMT\project.exe
Enter infix expression: A*B/C
Postfix expression: A B * C /

Process exited after 11.52 seconds with return value 0
Press any key to continue . . .
Type here to search
ENG
2:44 PM
5/19/2023


C:\Dev C\HTMT\project.exe
Enter infix expression: ((A+B)*C+D)/(E+F+G)
Postfix expression: A B + C * D + E F + G + /

Process exited after 20.06 seconds with return value 0
Press any key to continue . . .
Type here to search
ENG
2:44 PM
5/19/2023