

NAME: GAIRE ANANTA PRASAD

M24W0272

1. C Language code for Infix to Postfix.

```
1  /*GAIRE ANANTA PRASAD*/
2  /*M24W0272*/
3
4  #include <stdio.h>
5  #include <stdlib.h>
6  #include <ctype.h>
7
8  // Structure for stack
9  struct Stack {
10     int top;
11     unsigned capacity;
12     int* array;
13 };
14
15 // Function to create a stack
16 struct Stack* createStack(unsigned capacity) {
17     struct Stack* stack = (struct Stack*)malloc(sizeof(struct Stack));
18     stack->capacity = capacity;
19     stack->top = -1;
20     stack->array = (int*)malloc(stack->capacity * sizeof(int));
21     return stack;
22 }
23
24 // Function to check if the stack is empty
25 int isEmpty(struct Stack* stack) {
26     return stack->top == -1;
27 }
28
29 // Function to push element onto stack
30 void push(struct Stack* stack, int item) {
31     stack->array[++stack->top] = item;
32 }
33
34 // Function to pop element from stack
35 int pop(struct Stack* stack) {
36     if (!isEmpty(stack))
37         return stack->array[stack->top--];
38 }
39
40 // Function to return the precedence of an operator
41 int precedence(char op) {
42     if (op == '+' || op == '-')
43         return 1;
44     if (op == '*' || op == '/')
45         return 2;
46     return 0;
47 }
48
49 // Function to convert infix expression to postfix expression
50 void infixToPostfix(char* infix, char* postfix) {
51     struct Stack* stack = createStack(strlen(infix));
52     int i, k;
53     for (i = 0, k = -1; infix[i]; ++i) {
54         if (isalnum(infix[i]))
55             postfix[++k] = infix[i];
56         else if (infix[i] == '(')
57             push(stack, infix[i]);
58         else if (infix[i] == ')') {
59             while (!isEmpty(stack) && stack->array[stack->top] != '(')
60                 postfix[++k] = pop(stack);
61             if (!isEmpty(stack) && stack->array[stack->top] != '(')
62                 return; // Invalid expression
63             else
64                 pop(stack);
65         } else { // Operator
66             while (!isEmpty(stack) && precedence(infix[i]) <= precedence(stack->array[stack->top]))
67                 postfix[++k] = pop(stack);
68             push(stack, infix[i]);
69         }
70     }
```

GAIRE ANANTA PRASAD

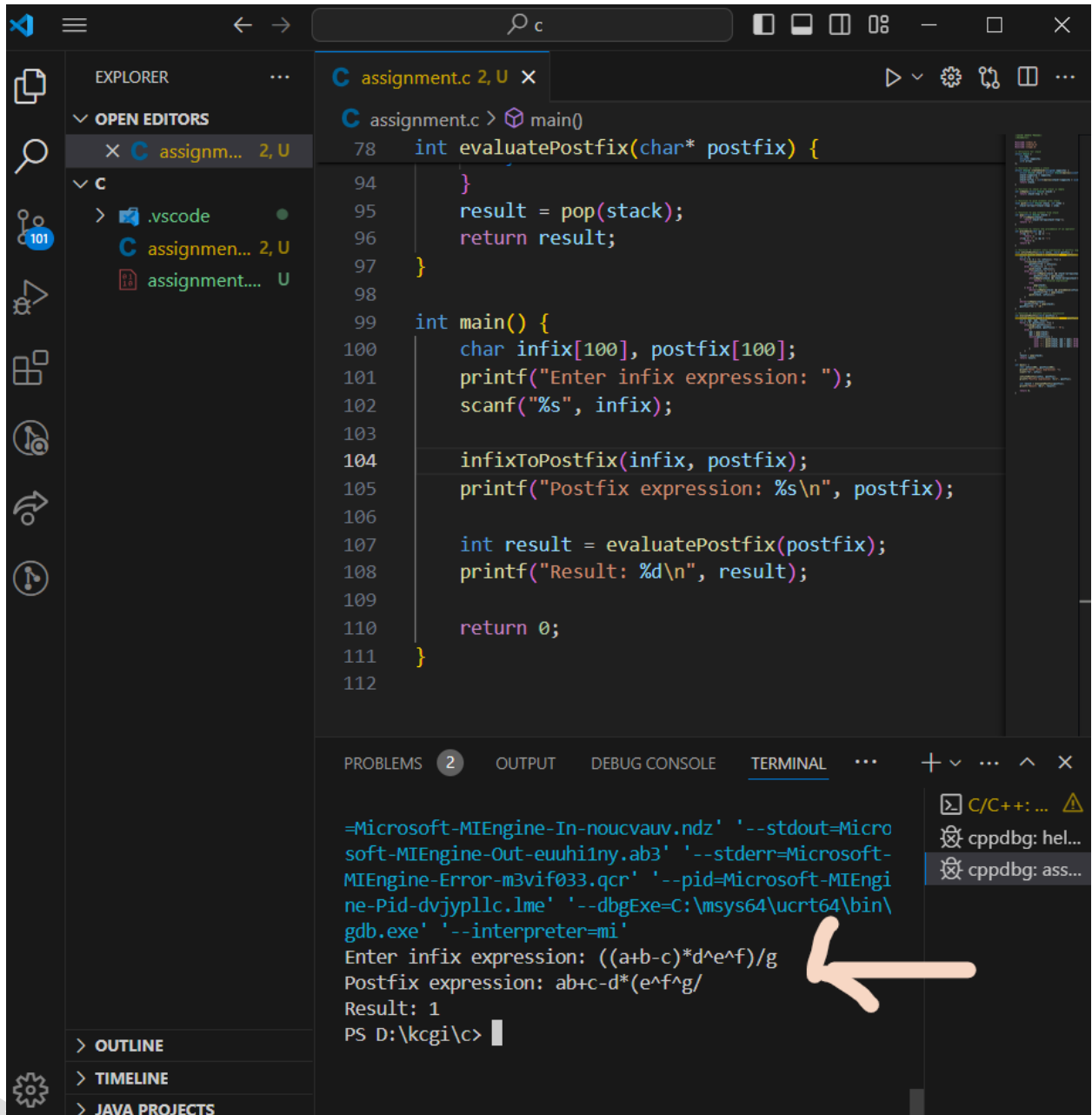
M24W0272

```
51 void infixToPostfix(char* infix, char* postfix) {
70 }
71
72 while(!isEmpty(stack))
73     postfix[++k] = pop(stack);
74 postfix[++k] = '\0';
75 }
76
77 // Function to evaluate postfix expression
78 int evaluatePostfix(char* postfix) {
79     struct Stack* stack = createStack(strlen(postfix));
80     int i, op1, op2, result;
81     for(i = 0; postfix[i]; ++i) {
82         if(isdigit(postfix[i]))
83             push(stack, postfix[i] - '0');
84         else {
85             op2 = pop(stack);
86             op1 = pop(stack);
87             switch(postfix[i]) {
88                 case '+': push(stack, op1 + op2); break;
89                 case '-': push(stack, op1 - op2); break;
90                 case '*': push(stack, op1 * op2); break;
91                 case '/': push(stack, op1 / op2); break;
92             }
93         }
94     }
95     result = pop(stack);
96     return result;
97 }
98
99 int main() {
100     char infix[100], postfix[100];
101     printf("Enter infix expression: ");
102     scanf("%s", infix);
103
104     infixToPostfix(infix, postfix);
105     printf("Postfix expression: %s\n", postfix);
106
107     int result = evaluatePostfix(postfix);
108     printf("Result: %d\n", result);
109
110     return 0;
111 }
```

```
78 int evaluatePostfix(char* postfix) {
92     }
93 }
94 }
95 result = pop(stack);
96 return result;
97 }
98
99 int main() {
100     char infix[100], postfix[100];
101     printf("Enter infix expression: ");
102     scanf("%s", infix);
103
104     infixToPostfix(infix, postfix);
105     printf("Postfix expression: %s\n", postfix);
106
107     int result = evaluatePostfix(postfix);
108     printf("Result: %d\n", result);
109
110     return 0;
111 }
```

GAIRE ANANTA PRASAD
M24W0272

2. Output



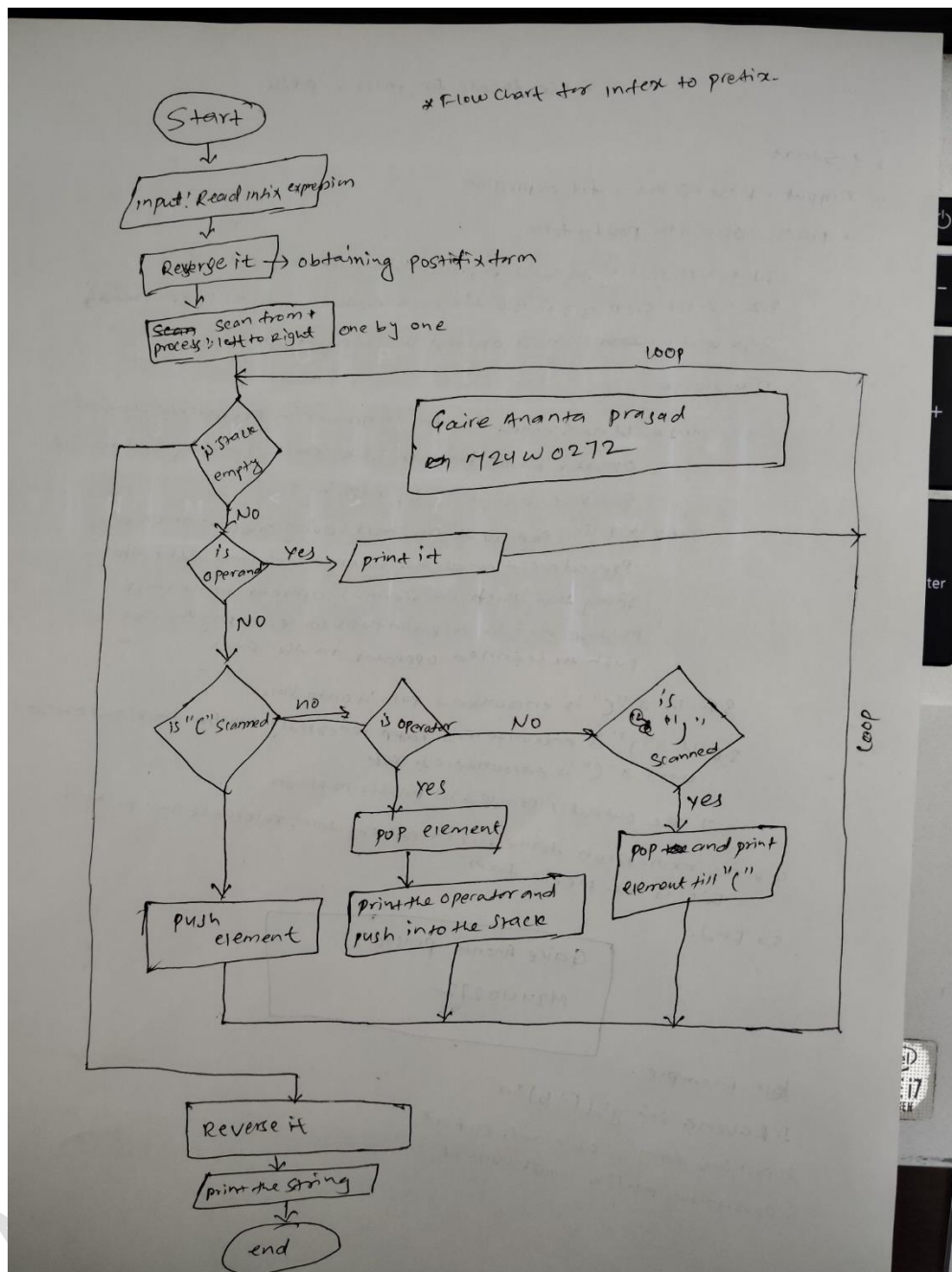
```
assignment.c > main()
78  int evaluatePostfix(char* postfix) {
94      }
95      result = pop(stack);
96      return result;
97  }
98
99  int main() {
100     char infix[100], postfix[100];
101     printf("Enter infix expression: ");
102     scanf("%s", infix);
103
104     infixToPostfix(infix, postfix);
105     printf("Postfix expression: %s\n", postfix);
106
107     int result = evaluatePostfix(postfix);
108     printf("Result: %d\n", result);
109
110     return 0;
111 }
112
```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL ...

=Microsoft-MIEngine-In-noucvaav.ndz' '--stdout=Micro
soft-MIEngine-Out-euuhi1ny.ab3' '--stderr=Microsoft-
MIEngine-Error-m3vif033.qcr' '--pid=Microsoft-MIEngi
ne-Pid-dvjypllc.lme' '--dbgExe=C:\msys64\ucrt64\bin\
gdb.exe' '--interpreter=mi'
Enter infix expression: ((a+b-c)*d^e^f)/g
Postfix expression: ab+c-d*(e^f^g/
Result: 1
PS D:\kcgi\c> |

C/C++: ...
cppdbg: hel...
cppdbg: ass...

3. Flow-Chart for Infix to Postfix.



4. Pseudocode for Infix to Postfix.

* pseudocode for infix to postfix.

- 1 * Start
- 2 * Input:- Reverse the infix expression
- 3 * process:- Obtain the postfix form
 - 3.1 * Start reading the infix expression from left to right
 - 3.2 * Repeat Step 3.3 to 3.6 for each element until the stack is empty.
 - 3.3 * If it is ~~scan~~ scan 21 operand we output it, print it.
 - 3.4 * Else
 - 3.4.1 * If the Scanned Operator is greater in precedence than the Operator in the Stack or if the Stack is empty or the Stack contains a "(", push it.
 - 3.4.2 * Else, Pop all the operators having greater or equal precedence than of the scanned operator. After that doing that Push the scanned operator to the stack. In case there is a parenthesis while popping then stop and push the scanned operator in the stack.
 - 3.5. If a "(" is encountered, push it onto stack.
 - 3.6 If a ")" is encountered, ~~stop~~ repeatedly pop from stack and output it until a "(" is encountered. ~~###~~
 - 3.7. The output is printed in postfix notation.
- 4 * The expression formed is in postfix form, reverse it and print it, this is the prefix form.
- 5 * End.

Gaire Ananta prasad
M24W0272

Ex Example:-

1. Reverse string: $(a+b)*a$
2. postfix form is obtained: $cb+a*$
3. Reverse the postfix result: $a+b*$