Lab 02

TASK: WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators + , - , *, /, ^

DATE: 06-10-25

Infix Expression: a+b*c-d/e+f*g/h^i
Postfix Expression: abc*+de/-fg*hi^/+

Infix Expression: (a+b*c/t(h*t/e-463))
Postfix Expression: abc*tht*e/463-/+

```
lab-03 06-10-25
wap to convert a given valid paranthe sized infra
arthmetic expression to post-fix expression. The
expression considers of a single character operands and
the binary operands +, -, *, 1, "
NINFIX TO POSTFIX
# Include LS+dio-n>
# Include ( ctype. h >
# Include LString.h>
#define MAX 100
 char stack [max];
 Put tob= -7;
 11 Push function
 voia push (chor c) {
     if (top= = MAX - 1) {
          Printf ("Stock Overflow");
            return;
     top+t;
     Stack[top] = C;
                              October 06, 2025 10:56 PM
```

```
1/ Pop function. Char pop () {
     it (fob = = -7) {
     Prent ("Stack Underflowin");
        Mind State Vider
                                            11 F
                                           110
    retors top 1;
                                            int
     return stack [to
    Char element-pop = stack[top];
    top -- ;
    return element-POPi
1/ Function to return precedence
int precedence (char op) {
  Switch (op) {
     case '+':
     case '-':
     return 1;
    Case '+';
     Case 1'
     return 2;
    Case 'n':
       return 3;
    Case 'C':
         return o;
                        October 06, 2025 10:56 PM
```

```
return - 2;
// function to return Associativity
11 0 = feft to right, 1 = Right to feft
int associativity (chor op) {

f(op == "1") { return 1;
   return 0; 11+, -, *, 1
1/ Function to convert infix to postfix
void infix -to-postfix (char infix[], char postfix[]){
   Int P, K = 0;
   char er
for (Pxt 9=0; Pnflx[i] 1= 10; i+1) {
       c: infix[i];
       if (isalnum(c)){
            11 If operand direct to post fix
            postfix[K++] = C;
                            October 06, 2025 10:56 PM
```

```
// Pop remaining operators.
While (top != -1) {
                 postfix [K++] = Popes;
            gost fix [k] = "(0")
            int main () {
             char infex [MAX], postfex [MAX];
             printf("Infix Expression: ");
             Sconf (" " o's", in fix);
             infix-to-Postfix (9nfix, Postfix);
              Prints ("Postfix Expression: %s/n", Postfix);
             return o',
ice(cs) 1)
ice (c) #
-R.
            OUTPUT :-
            Infix Expression: 4+B|C*D NAMED
            AXBANAM
            Postfix Expression: ABC | D* +
            Jufax Expression = (A+(B*C-(D)E^F)*G) + H)
            Postfex Expression: ABC * DEF 1) G * - H*+
                                       October 06, 2025 10:56 PM
```

```
#include<stdio.h>
     #include<ctype.h>
     #include<string.h>
     #define MAX 100
    void push(char c);
     char pop();
     char peek();
10
     int associativity(char operator);
11
12
     int precedence(char operator);
13
     void infix_to_postfix(char infix[], char postfix[]);
14
15
16
    char infix[MAX];
17
    char postfix[MAX];
18
    char stack[MAX];
19
    int top = -1;
20
     int k = 0;
21
22
     int main(void){
23
24
         printf("Infix Expression: ");
25
         scanf("%s", infix);
```

```
infix_to_postfix(infix, postfix);
27
28
         printf("Postfix Expression: ");
29
         printf("%s", postfix);
30
31
32
33
     void push(char element){
         if(top == MAX-1) return;
34
35
         top++;
         stack[top] = element;
36
37
38
     char pop(){
39
         char element = stack[top];
40
41
         top--;
42
         return element;
43
44
45
     char peek(){
46
         if (top == -1) return '\0';
47
         return stack[top];
48
49
```

```
int precedence(char op) {
51
52
         switch(op) {
             case '+':
53
54
             case '-':
55
               return 1;
56
             case '*':
57
             case '/':
58
                 return 2;
             case '^':
59
60
                 return 3;
             default:
61
                 return -1; // for non-operators
62
63
64
     }
65
66
67
     int associativity(char op){
         switch(op){
68
             case '+':
69
             case '-':
70
71
             case '*':
72
             case '/':
73
                 return 1;
74
             case '^':
75
                 return 2;
             default:
76
77
                 return -1;
78
79
     }
80
81
```

```
void infix_to_postfix(char infix[], char postfix[]){
          for(int i = 0; i < strlen(infix); i++){</pre>
              char incoming = infix[i];
              if (incoming == '(') push(incoming);
else if (isalnum(incoming)) postfix[k++] = incoming;
              else if (incoming == ')'){
                  while((top != -1) && (peek() != '(')) postfix[k++] = pop();
                   pop();//to remove the (
              else if (precedence(incoming) > precedence(peek())) push(incoming);
                   while (top != -1 && ((precedence(incoming) < precedence(peek())) ||
                   (precedence(incoming) == precedence(peek()) && associativity(incoming) == 1)))
98
                       postfix[k++] = pop();
103
104
                   push(incoming);
105
107
108
109
          while(top != -1) postfix[k++] = pop();
110
111
          postfix[k++] = '\0';
112
113
```

TASK: Check for balanced parentheses

DATE: 06-09-25

Expression: {{[[()

Unbalanced

Expression: ([({{[]}})])

Balanced

Expression: [(this_is_an_example)]

Balanced

```
Lab 02 Balancing Parantheses. 29/09/25
# Include ( stdio. h)
# Include (String. h)
# define MAX 100
char stack [MAX] / Anton . May 10 = 3 +11
int top = -1;
void push (char ?);
void pop();
char peck ();
and match (char top, chor incoming);
int mainer {
     char expression[MAx];
     printf( Expression: ");
     scanti 1. s", 4 expression);
   for (Put 9=0; 92 Strlen (expression); i++) {
          of (expression [i] = = "(" H)
             expression [1] = = "{" 11
             ex pression Ci] = = "["]{
             push (expression [i]); remaind
                          October 06, 2025 11:21 PM
```

```
if (expression (1) == ")" |
           expression[1] == "] War others a pulsaries
           expression (P) == "J') { prepries a laboration
            char stack.top = peekc);
           int i = check-match ( stack-top, experiency)
           of (i==1) f
                poper)
          else {

printf("Un balanced");
                                POSTON Mi
if (top ===1) { [ [ ] | 1 | (0 = 1 | 1 ) ] ] ].
       printf(" & Balanced");
return of 1007 adapting 1 1344.
                              October 06, 2025 11:21 PM
```

```
void Push (char i) {
           if (top = = MAX -1){
                 Printse" Stack Overflow");
           etse {
             top++;
, City
              Stack [top] = ";
              ********
    vord pop ( ) {
        if (top = = -1) {

printf(" underflow");
       else {
        return stack [top];
                                 October 06, 2025 11:21 PM
```

```
Int match (ther top, that incoming) {
                                                      Lab-
                                                     WAP
                                                     arfth
                                                      expr
           return 1;
                                                      the
                                                      11]
                                                       #
     return o'
Output! - (Cuellinghay "12 taken
Expression: [ } }]
Balanced.
Expression: [[]]
Un Balanced
                                October 06, 2025 11:21 PM
```

```
#include <stdio.h>
      #include <string.h>
      #define MAX 100
      char stack[MAX];
      int top = -1;
      void push(char i);
      void pop();
char peek();
int match(char top, char incoming);
      int main(){
           char expression[MAX];
printf("Expression: ");
scanf("%s", &expression);
           for(int i = 0; i < strlen(expression); i++)[]
  if (expression[i] == '(' || expression[i] == '{' || expression[i] == '['){</pre>
19
                     push(expression[i]);
                if (expression[i] == ')' || expression[i] == '}' || expression[i] == ']'){
                     char stack_top = peek();
                     int j = match(stack_top, expression[i]);
                     if (j == 1){
                          pop();
                     else{
                          printf("Unbalanced");
```

```
if (top == -1){}
39
             printf("Balanced");
40
41
42
43
         return 0;
44
45
     //this is shrihari viswanahan program'
46
47
48
     void push(char i){
49
         if (top == MAX - 1){
             printf("Stack Overflow");
50
51
52
         else{
53
             top++;
             stack[top] = i;
54
55
56
57
58
     void pop(){
         if (top == -1){
59
             printf("Stack underflow");
60
61
         else{
62
63
             int item = stack[top];
64
             top--;
65
66
67
```

```
char peek(){
68
         if (top == -1){}
69
             printf("Underflow");
70
71
         else{
72
            return stack[top];
73
74
75
76
     int matc char top , char incoming){
77
         if (top == '(' && incoming == ')'){
78
79
            return 1;
         }
80
81
         if (top == '{' && incoming == '}'){
82
83
            return 1;
84
         }
85
         if (top == '[' && incoming == ']'){
86
87
            return 1;
88
89
90
         return 0;
91
92
```