

# VARUN BHARGAVA - 241010282

## DATA STRUCTURES TASK-6

### Task 01: Postfix to Prefix:

(<https://github.com/varunnnb/dsa-sem3-iiitnr/blob/main/lab6/lab6-1.c>)

Example: Convert the following Postfix expression into Prefix expression:

AB + CD - \*

Prefix expression:

\* + AB - CD

Question:

Convert the following Postfix expression into Prefix using stack operations:

P Q + RS \* T U - +V W/\*

**SOLUTION:**

++PQ\*+\*RS-TU-+VW/VW

The screenshot shows a terminal window titled "Code-lab6" with the command "cd" entered. The user then enters the postfix expression "PQ+RS\*TU-VW/\*" and receives the prefix expression "++PQ\*+\*RS-TU-+VW/VW" as output. The terminal also shows the path "C:\Users\varun\Desktop\V8\College\IIITNR\assignments\sem3\dsa\lab6".

```
cd "c:\Users\varun\Desktop\V8\College\IIITNR\assignments\sem3\dsa\lab6"; if ($?) { gcc lab6-1.c & lab6-1 }; if ($?) { .\lab6-1
enter postfix expression: PQ+RS*TU-VW/*
prefix expression: ++PQ*+*RS-TU-+VW/VW
PS C:\Users\varun\Desktop\V8\College\IIITNR\assignments\sem3\dsa\lab6>
```

```
lab6 > C lab6-1.c > push(char*)
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 char stack[100][100];
6 int top = -1;
7
8 void push(char *str)
9 {
10     if (top < 99)
11     {
12         strcpy(stack[++top], str);
13     }
14     else
15     {
16         printf("overflow\n");
17     }
18 }
19
20 char *pop()
21 {
22     if (top >= 0)
23     {
24         return stack[top--];
25     }
26     else
27     {
28         printf("underflow\n");
29         return 0;
30     }
31 }
32
33 int isOperator(char c)
34 {
35     return (c == '+') || (c == '-') || (c == '**') || (c == '/');
36 }
37
38 int isOperand(char c)
39 {
40     return ((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z'));
41 }
```

```

lab6 > C:\lab6-1.c > push(char)
43 char toLowerCase(char c)
44 {
45     if (c >= 'A' && c <= 'Z')
46     {
47         return c + 32;
48     }
49     return c;
50 }
51
52 int main()
53 {
54     char postfix[100];
55     printf("Enter postfix expression: ");
56     fgets(postfix, sizeof(postfix), stdin);
57
58     postfix[strcspn(postfix, "\n")] = 0;
59
60     for (int i = 0; i < strlen(postfix); i++)
61     {
62         char c = postfix[i];
63
64         if (c == ' ' || c == '\t')
65             continue;
66
67         if (isOperand(c))
68         {
69             char op[2] = {toLowerCase(c), '\0'};
70             push(op);
71         }
72         else if (isOperator(c))
73         {
74             char op1[100], op2[100], expr[100];
75             strcpy(op2, pop());
76             strcpy(op1, pop());
77             sprintf(expr, "%s%s%s", c, op1, op2);
78             push(expr);
79         }
80     }
81
82     printf("prefix expression: %s\n", stack[top]);
83 }

```

cd "c:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab6"; if (\$?) { gcc lab6-1.c -o lab6-1 } ; if (\$?) { ./lab6-1 }
enter postfix expression: PQ+RS\*10-+W/+V
prefix expression: +PQ+\*RS-10+W+V

## Task 02: Stack-based Parentheses Checker:

( <https://github.com/varunnnb/dsa-sem3-iiitnr/blob/main/lab6/lab6-2.c> )

Write a program using a stack to check whether a given expression has balanced parentheses or not. The expression may include the symbols: (), {}, and [].

- Display “Balanced” if all parentheses are matched properly, otherwise “Not Balanced”.

Examples:

- Input: { [ ( A + B ) \* ( C + D ) ] }

Output: Balanced

- Input: { ( A + B ) }

Output: Not Balanced

Expressions to check:

1. { [ ( A + B ) \* ( C + D ) ) }

2. ( A + B ] { C + D }

3. { ( [ A + B ] \* C ) - D }

### SOLUTION:

1. NOT BALANCED

## 2. NOT BALANCED

The screenshot shows a terminal window with the following command and output:

```
cd "c:\Users\varun\Desktop\VB\College\IITNR\assignments\sem3\dsa" ; if ($?) { gcc lab6-2.c -o lab6-2 } ; if (?) { ./lab6-2
> enter expression: ( A + B ] ) C + D
not balanced
PS C:\Users\varun\Desktop\VB\College\IITNR\assignments\sem3\dsa
```

### 3. BALANCED

The screenshot shows a terminal window with the following content:

```
● > cd "c:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\" ; if ($?) { gcc lab6-2.c -o lab6-2 } ; if ($?) { ./lab6-2 }
} enter expression: { ( [ A + B ] * C ) - D }
balanced
○ PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa>
```

The terminal window is part of a larger interface, likely a code editor or terminal emulator, which also displays the source code of the C program `lab6-2.c`. The code is as follows:

```
42 int checkBalanced(char expr[])
43 {
44     for (int i = 0; i < strlen(expr); i++)
45     {
46         if (top == -1)
47             return 1;
48         else
49             return 0;
50     }
51
52     int main()
53     {
54         char expr[100];
55
56         printf("enter expression: ");
57         fgets(expr, sizeof(expr), stdin);
58         expr[strcspn(expr, "\n")] = 0;
59
60         if (checkBalanced(expr))
61         {
62             printf("balanced\n");
63         }
64         else
65         {
66             printf("not balanced\n");
67         }
68
69         return 0;
70     }
71 }
```