Postfix to Infix Conversion

Algorithm of Postfix to Infix

Expression = abc-+de-fg-h+/*

```
1.While there are input symbol left
2.      Read the next symbol from input.
3.      If the symbol is an operand
                Push it onto the stack.
4.      Otherwise,
                the symbol is an operator.
5.      If there are fewer than 2 values on the stack
                Show Error /* input not sufficient values in the expression */
6.      Else
                Pop the top 2 values from the stack.
                Put the operator, with the values as arguments and form a string.
                Encapsulate the resulted string with parenthesis.
                Push the resulted string back to stack.
7.      If there is only one value in the stack
                That value in the stack is the desired infix string.
8.      If there are more values in the stack
                Show Error /* The user input has too many values */
```

- Prefix Infix Postfix converter Tool Online

Postfix to Infix conversion

Example

```
abc-+de-fg-h+/*
```

Expression	Stack
abc-+de-fg-h+/*	NuLL
bc-+de-fg-h+/*	"a"
c-+de-fg-h+/*	"b" "a"
-+de-fg-h+/*	"c" "b" "a"
+de-fg-h+/*	"b - c" "a"
de-fg-h+/*	"a+b-c"
e-fg-h+/*	"d" "a+b-c"
-fg-h+/*	"e" "d" "a+b-c"
fg-h+/*	"d - e" "a+b-c"
g-h+/*	"f" "d - e" "a+b-c"
-h+/*	"g" "f" "d - e" "a+b-c"
h+/*	"f-g" "d - e" "a+b-c"
+/*	"h" "f-g" "d - e"

	"a+b-c"
/*	"f-g+h" "d - e" "a+b-c"
*	"(d-e)/(f-g-h)" "a+b-c"
Null	"(a+b-c)*(d-e)/(f-g+h)"

Ans = (a+b-c)*(d-e)/(f-g+h)

Postfix to Infix implementation in c

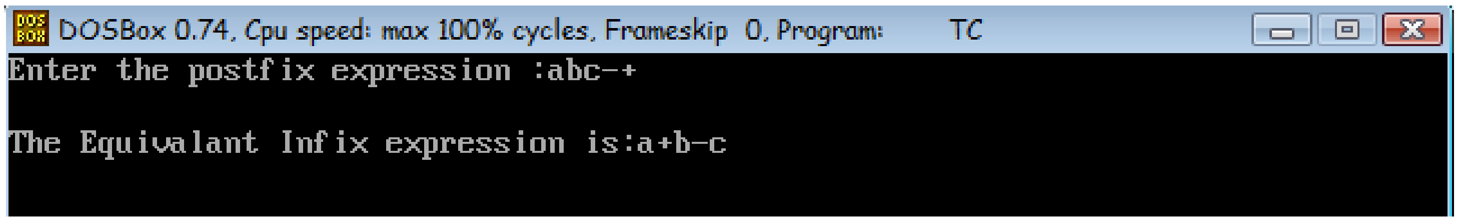
```
#include <stdio.h>
#include <stdlib.h>
int top = 10;
struct node
{
        char ch;
        struct node *next;
        struct node *prev;
}  *stack[11];
typedef struct node node;

void push(node *str)
{
        if (top <= 0)
        printf("Stack is Full ");
        else
        {
                stack[top] = str;
                top--;
        }
}

node *pop()
{
        node *exp;
        if (top >= 10)
                printf("Stack is Empty ");
        else
                exp = stack[++top];
        return exp;
}
void convert(char exp[])
{
        node *op1,  *op2;
        node *temp;
        int i;
        for (i=0;exp[i]!='\0';i++)
        if (exp[i] >= 'a'&& exp[i] <= 'z'|| exp[i] >= 'A' && exp[i] <= 'Z')
        {
                temp = (node*)malloc(sizeof(node));
                temp->ch = exp[i];
                temp->next = NULL;
                temp->prev = NULL;
                push(temp);
        }
        else if (exp[i] == '+' || exp[i] == '-' || exp[i] == '*' || exp[i] == '/' ||
exp[i] == '^')
        {
                op1 = pop();
                op2 = pop();
                temp = (node*)malloc(sizeof(node));
                temp->ch = exp[i];
                temp->next = op1;
                temp->prev = op2;
                push(temp);
        }
}

void display(node *temp)
{
        if (temp != NULL)
        {
                display(temp->prev);
                printf("%c", temp->ch);
                display(temp->next);
        }
}

void main()
{
        char exp[50];
        clrscr();
        printf("Enter the postfix expression :");
        scanf("%s", exp);
        convert(exp);
        printf("\nThe Equivalant Infix expression is:");
        display(pop());
        printf("\n\n");
        getch();
}
```


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
Enter the postfix expression :abc-+

The Equivalant Infix expression is:a+b-c
```