Infix to Postfix (LAB-3)

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

#define MAX 100
char stack[MAX];
int top = -1;

void push(char);
char pop();
int precedence(char);
void infixtopostfix(char infix[], char postfix[]);

void push(char x)
{
    if (top == MAX-1)
    {
        printf("overflow!\n");
        exit(EXIT_FAILURE);
    }
    else
    {
        top++;
        stack[top] = x;
    }
}

char pop()
{
    if (top == -1)
    {
        printf("underflow\n");
    }
    else
    {
        char popped = stack[top];
        top--;
```

```
        return popped;
    }
}
int precedence (char symbol)
{
    if (symbol=='^')
        return 3;
    else if (symbol=='*' || symbol=='/')
        return 2;
    else if (symbol=='+' || symbol=='-')
        return 1;
    else
        return -1;
}
void infix to postfix (char infix[], char postfix[])
{
    int i=0, j=0;
    char symbol, temp;
    push('#');
    while ((symbol= infix[i++])!='\0')
    {
        if (symbol=='(')
            push(symbol);
        else if (isalnum(symbol))
            postfix[j++]=symbol;
        else if (symbol==')')
        {
            while(stack[top]!='(')
            {
                postfix[j++]=pop();
            }
            temp=pop();
        }
```

```
        else
        {
          while(precedence(stack[top]) >= precedence(symbol))
          { postfix[j++] = pop();
          }
          push(symbol);
        }
    }
    while(stack[top] != '#')
    {
      postfix[j++] = pop();
    }
    postfix[j] = '\0';
}

int main()
{
  char infix[MAX], postfix[MAX];
  printf("Enter a valid parenthesized infix exp:\n");
  scanf("%s", infix);
  infixtopostfix(infix, postfix);
  printf("The postfix exp is: %s\n", postfix);
  return 0;
}
```

Output:

Enter a valid parenthesized infix exp:

a*b+c*d-e

The postfix exp is: ab*cd*+e-

28/12/2023