LAB-2

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <stdlib.h>
#define SIZE 50

char stack[SIZE];
int top = -1;

void push(char elem) {
    if (top == SIZE-1)
        printf("Stack overflow");
    else
        stack[++top] = elem;
}
char pop() {
    if (top == -1)
        printf("Stack underflow");
    else {
        int popele = stack[top--];
        return popele;
    }
}

int pr(char symbol) {
    if (symbol == '^')
        return (3);
    else if (symbol == '*' || symbol == '/')
        return (2);
    else if (symbol == '+' || symbol == '-')
        return (1);
```

```
    else
        return (0);
}

int main () {
    char infix[50], postfix[50], ch, elem;
    int i = 0, k = 0;
    printf ("Enter infix expression: ");
    scanf ("%s", infix);
    for(i=0; i < strlen (infix); ++i){
        if (( infix [i] == '*' || infix [i] == '+' || infix [i] == '/' ||
            infix [i] == '-' || infix [i] == '^' || infix [i] == '(' ) &&
            infix [i+1] == '*' || infix [i+1] == '+' || infix [i+1] == '/' ||
            infix [i+1] == '-' || infix [i+1] == '^' || infix [i+1] == ')' )) {
                printf (" Invalid Expression");
                exit (1);
        }
    }
    push ('#');
    while ((ch = infix [i++]) != '\0') {
        if (ch == '(')
        push (ch);
        else
        if (isalnum(ch)) postfix [k++] = ch;
            else
                if (ch == ')') {
                    while (stack [top] != '(')
                        postfix [k++] = pop();
                    elem = pop();
                }
```

```
        else {
            while (pr(stack[top]) >= pr(ch))
                postfix[k++] = pop();
            push(ch);
        }
    }
    while (stack[top] != '#')
        postfix[k++] = pop();

    postfix[k] = '\0';
    printf("\n Postfix Expression = %s\n", postfix);

    return 0;
}
```