LAB PROGRAM 2

WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression.

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

#define MAX 5
char stack[MAX];
int top = -1;

void push(char c) {
   if (top < MAX - 1) {
      stack[++top] = c;
   }
}

char pop() {
   if (top >= 0) {
      return stack[top--];
   }
   return '\0';
}

char peek() {
   if (top >= 0) {
      return stack[top];
   }
   return '\0';
}
```

```
int precedence(char c) {
    switch (c) {
        case '+': return 1;
        case '-': return 1;
        case '*': return 2;
        case '/': return 2;
        case '^': return 3;
        default: return 0;
    }
}

int isOperator(char c) {
    return c == '+' || c == '-' || c == '*' || c == '/'|| c=='^';
}

void infixToPostfix(const char *infix, char *postfix) {
    int i = 0, j = 0;
    while (infix[i]) {
        if (isalnum(infix[i])) {
            postfix[j++] = infix[i];
        } else if (infix[i] == '(') {
            push(infix[i]);
        } else if (infix[i] == ')') {
            while (top != -1 && peek() != '(') {
                postfix[j++] = pop();
            }
            pop();
        } else if (isOperator(infix[i])) {
            while (top != -1 && precedence(peek()) >= precedence(infix[i])) {
                postfix[j++] = pop();
            }
            push(infix[i]);
        }
        i++;
    }
    while (top != -1) {
        postfix[j++] = pop();
    }
    postfix[j] = '\0';
}

int main() {
    char infix[MAX], postfix[MAX];

    printf("Enter an infix expression: ");
    scanf("%s", infix);

    infixToPostfix(infix, postfix);
    printf("Postfix expression: %s\n", postfix);

    return 0;
}
```