postfix.c

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

#define MAX 100
char stack[MAX];
int top=-1;
void push(char ch)
{
    if(top==MAX-1)
    {
        printf("overflow");
        return;
    }
    stack[++top]=ch;
}
char pop()
{
    if(top==-1)
    {
        printf("empty");
        return'\0';
    }
    return stack[top--];
}
char peek()
{
    return stack[top];
}
int precedence(char op)
{
    if(op=='^') return 3;
    if (op == '*' || op == '/') return 2;
    if (op == '+' || op == '-') return 1;
    return 0;
}
```

postfix.c

```c
int isOperator(char ch)
{
    return ch == '+' || ch == '-' || ch == '*' || ch == '/' || ch == '^';
}
void infixToPostfix(char *infix,char *postfix)
{
    int i,k=0;
    for (i = 0; infix[i] != '\0'; i++) {
        char ch = infix[i];
        if(isalnum(ch))
        {
            postfix[k++]=ch;
        }
        else if(ch=='(')
        {
            push(ch);
        }
        else if(ch==')')
        {
            while (peek()!='(')
            {
                postfix[k++]=pop();
            }
            pop();
        }
        else if(isOperator(ch))
        {
            while (precedence(peek())>=precedence(ch))
            {
                postfix[k++]=pop();
            }
            push(ch);

        }
    }
    while (top != -1)
```

```
    }
    while (top != -1)
        postfix[k++] = pop();

    postfix[k] = '\0';
}
int main() {
    char infix[MAX], postfix[MAX];
    printf("Enter infix expression: ");
    scanf("%s", infix);

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

    return 0;
}
```