

Infix to Postfix

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

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

void push(char c){
    if(top == MAX -1){
        printf("Stack Overflow\n");
    }else{
        top = top+1;
        stack[top] = c;
    }
}

char pop(){
    char val;
    if(top == -1){
        printf("Stack Underflow\n");
        return -1;
    }else{
        val = stack[top];
        top = top-1;
        return val;
    }
}
```

```

char peek(){

    if(top == -1)
        return '\0';

    return stack[top];
}

int precedence(char c){

    if(c == '+' || c == '-')
        return 1;

    if(c == '*' || c == '/')
        return 2;

    if(c == '^' )
        return 3;

    return 0;
}

void infixToPostfix(char infix[], char postfix[]){

    int i,k=0;

    char c;

    for(i=0;infix[i]!='\0'; i++){
        c = infix[i];

        if(isalnum(c)){
            postfix[k] = c;
            k=k+1;
        }

        else if(c=='('){
            push(c);
        }

        else if(c == ')'){
            while(top!=-1 && peek() != '('){

                postfix[k] = pop();
            }
        }
    }
}

```

```

        k = k+1;

    }

    pop();

}

else{

    while(top!=-1 && precedence(peek()) >= precedence(c)){

        postfix[k] = pop(c);

        k=k+1;

    }

    push(c);

}

}

while(top!=-1){

    postfix[k] = pop();

    k=k+1;

}

postfix[k] = '\0';

}

int main(){

char infix[MAX] , postfix[MAX];

printf("Enter a valid parenthesized infix expression:");

scanf("%s", infix);

infixToPostfix(infix,postfix);

printf("Postfix Expression : %s\n", postfix);

return 0;

}

```

Output:

```
Enter a valid parenthesized infix expression:(2+3-1(6/3+(4-3)*5))
Postfix Expression : 23+163/43-5*+-
```