6_1_infix_to_postfix.c

C: > Users > shuvr > OneDrive > Documents > CODING > College C codes > DSA-ASS-6 > 6_1_infix_to_postfix.c > ...

```c
/* Note: Input should be taken from the user
Write a C/C++ program to convert an infix expression to a postfix expression using a stack, handling the operators +, -, *, /, and parentheses.
Example :--
Input: A*(B+C*D)+E
Output: ABCD*+*E+ */

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

#define MAX 100

struct stack{
    char items[MAX];
    int top;
};

void initStack(struct stack *s){
    s->top=-1;
}

int isEmpty(struct stack *s){
    return s->top==-1;
}

void push(struct stack *s, char c){
    if(s->top<MAX-1){
        s->items[++(s->top)]=c;
    }
}

char pop(struct stack *s){
    if(!isEmpty(s)){
        return s->items[(s->top)--];
    }
    return '\0';
}

char peek(struct stack *s){
    if(!isEmpty(s)){
        return s->items[s->top];
    }
    return '\0';
}

```

```c

int precedence(char op){
    switch (op){
        case '+':            // fall through and no break required because of return statement
        case '-': return 1;
        case '*':
        case '/': return 2;
        case '^': return 3;
        default: return 0;
    }
}

void infixToPostfix(char *infix, char *postfix){
    struct stack *s=(struct stack *)malloc(sizeof(struct stack));
    initStack(s);
    int j=0;
    for(int i=0;infix[i]!='\0';i++){
        char c=infix[i];
        if(isdigit(c) || isalpha(c)){
            postfix[j++]=c;
        }
        else if(c=='('){
            push(s,c);
        }
        else if(c==')'){
            while(!isEmpty(s) && peek(s)!='('){
                postfix[j++]=pop(s);
            }
            pop(s); //to removee '('
        }
        else{
            while(!isEmpty(s) && precedence(peek(s))>=precedence(c)){
                postfix[j++]=pop(s);
            }
            push(s,c);
        }
    }
    while(!isEmpty(s)){
        postfix[j++]=pop(s);
    }
    postfix[j]='\0';
}

```

```c
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;
}
```

TERMINAL COMMENTS

```
Enter Infix Expression : A*(B+C*D)+E
Postfix Expression : ABCD*+*E+
```