```c
#include<stdio.h>
#include<string.h>
#define size 30
char stack[size], infix[size], postfix[size];
int top=-1;
void push(char symbol){
    if(top == size-1){
        printf("Stack is already full !");
        return;
    }
    stack[++top] = symbol;
}
char pop(){
    if(top == -1)
        return '!';
    return stack[top--];
}
char peek(){
    if(top == -1)
        return '!';
    return stack[top];
}

int weightage(char sign){
    switch (sign) {
    case '+':
    case '-':
        return 1;
        break;    case '*':
    case '/':
    case '%':
        return 2;
        break;
    case '^':
        return 3;
        break;
     default:
        return -1;
        break;
    }
}
void toPostfix(char inFix[], char post[]){
    char ch, extra;
    for(int i=0; i<strlen(inFix); i++){
        ch = inFix[i];
        if( (ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z') || ch == ' ')
            strncat(post,&ch,1);
        else if( inFix[i] == '(')
            push(ch);
```

```
        else if( inFix[i] == ')' ){
            while(peek() != '(' && top != -1){
                extra = pop();
                strncat(post,&extra,1);
            }
            if(top !=-1 && peek() != '(')
                printf("Invalid expression!");
            else
                pop();
        }
        else{
            while(top != -1 && weightage(ch) <= weightage(peek())){
                extra = pop();
                strncat(post,&extra,1);
            }
        push(ch);
        }
    } }
int main(){
    int choice = 1;
    while (choice == 1) {
        top = -1;
        printf("Enter the infix expression: ");
        gets(infix);
        strcat(infix,")");
        push('(');
        printf("Entered expression: %s\n",infix);
        toPostfix(infix, postfix);
        printf("Postfix expression: %s\n",postfix);
   printf("Do you want to find postfix of any other expression? Yes - 1,  no - 0\n");
        scanf("%d",&choice);
        gets(postfix);
    }
    printf("Bye!\n");
    return 0;
}
```

```
{ .\infixToPostMine }
Enter the infix expression: A+B*C/(E-F)
Entered expression: A+B*C/(E-F))
Postfix expression: ABC*EF-/+
Do you want to find postfix of any other expression? Yes - 1,  no - 0
1
Enter the infix expression: (A^Q-B)*(C+D)
Entered expression: (A^Q-B)*(C+D))
Postfix expression: AQ^B-CD+*
Do you want to find postfix of any other expression? Yes - 1,  no - 0
1
Enter the infix expression: a+b*(c^d-e)^(f+g*h)-i
Entered expression: a+b*(c^d-e)^(f+g*h)-i)
Postfix expression: abcd^e-fgh*+^*+i-
Do you want to find postfix of any other expression? Yes - 1,  no - 0
0
Bye!
```