INFIX TO PREFIX

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#define MAX 100
char stack[MAX];
int top=-1;
void push(char c) {
   if (top== MAX-1){
      printf("Stack Overflow\n");
   } else {
      stack[++top] = c;
   }
}
char pop(){
   if (top==-1) {
      return -1;
   } else {
      return stack[top--];
   }
}
int precedence(char c){
   if (c=='^')
      return 3;
   else if (c=='*'||c=='/')
      return 2;
   else if (c=='+'||c=='-')
      return 1;
   else
      return 0;
}
int isOperator(char c){
   return (c=='+'||c=='-'||c=='*'||c=='/'||c=='^');
}
void reverse(char *exp){
   int n = strlen(exp);
   for (int i=0;i<n / 2;i++) {
      char temp = exp[i];
      exp[i]=exp[n-i-1];
      exp[n-i-1]=temp;
   }
}
void infixToPrefix(char infix[], char prefix[]){
   char temp[MAX];
   int i,j=0;
   reverse(infix);
```

```
    for (i=0;infix[i]!='\0';i++){
        char symbol = infix[i];
        if (isalnum(symbol)){
            temp[j++]=symbol;
        }
        else if (symbol==')'){
            push(symbol);
        }
        else if (symbol=='('){
            while (top!=-1&&stack[top]!=')'){
                temp[j++] = pop();
            }
            pop();
        }
        else if (isOperator(symbol)){
            while (top != -1 && precedence(stack[top]) > precedence(symbol)) {
                temp[j++] = pop();
            }
            push(symbol);
        }
    }
    while (top != -1){
        temp[j++] = pop();
    }
    temp[j] ='\0';
    reverse(temp);
    strcpy(prefix,temp);
}
int main() {
    char infix[MAX], prefix[MAX];
    printf("Enter infix expression: ");
    gets(infix);
    infixToPrefix(infix, prefix);
    printf("Prefix Expression: %s\n", prefix);
    return 0;
}
```

OUTPUT:

```
Enter infix expression: (A-B/C)*(A/K-L)
Prefix Expression: *-A/BC-/AKL

--------------------------------
Process exited after 16.89 seconds with return value 0
Press any key to continue . . .
```