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

#define MAX 25

char stack[MAX];
int top = -1;

void push(char item)
{
    if(top >= MAX-1)
    {
        printf("\nStack Overflow.");
    }
    else
    {
        top = top+1;
        stack[top] = item;
    }
}

char pop()
{
    char item ;

    if(top == -1)
        return -1;
    else
    {
        item = stack[top];
        top = top-1;
        return(item);
    }
}


int check_operator(char symbol)
{
    if(symbol == '^' || symbol == '*' || symbol == '/' || symbol == '+' || symbol =='-')
        return 1;
```

```
    else
        return 0;
}

int precedence(char symbol)
{
    if(symbol == '^')
        return(3);
    else if(symbol == '*' || symbol == '/')
        return(2);
    else if(symbol == '+' || symbol == '-')
        return(1);
    else
        return(0);
}

void convert(char infix[], char postfix[])
{
    int i=0, j=0;
    char item;
    char x;

    push('(');
    strcat(infix,")");

    item=infix[i];

    while(item != '\0')
    {
        if(item == '(')
        {
            push(item);
        }
        else if( isdigit(item) || isalpha(item))
        {
         postfix[j] = item;
            j++;
        }
        else if(check_operator(item) == 1)
        {
            x=pop();
            while(check operator(x) == 1 && precedence(x)>= precedence(item))
```

```
85              {
86               postfix[j] = x;
87                  j++;
88                  x = pop();
89              }
90              push(x);
91              push(item);
92          }
93          else if(item == ')')
94          {
95              x = pop();
96              while(x != '(')
97              {
98               postfix[j] = x;
99                  j++;
100                 x = pop();
101             }
102         }
103         else
104         {
105             printf("\nInvalid infix Expression.\n");
106             getchar();
107             exit(1);
108         }
109         i++;
110         item = infix[i];
111     }
112     if(top>0)
113     {
114         printf("\nInvalid infix Expression.\n");
115         getchar();
116         exit(1);
117     }
118     if(top>0)
119     {
120         printf("\nInvalid infix Expression.\n");
121         getchar();
122         exit(1);
123     }
124
125
126 postfix[j] = '\0';
```

```
127 }
128
129
130 int main(int argc,char** argv)
131 {
132     char infix[MAX], postfix[MAX];
133     printf("Enter Infix expression : ");
134     gets(infix);
135
136  convert(infix,postfix);
137     printf("Postfix Expression: ");
138     puts(postfix);
139
140     return 0;
141 }
```