```
#include<stdio.h>
#include<string.h>
#define MAX 20

void intoprefix(char infix[20], char prefix[20]);
void reverse(char array[30]);
char pop();
void push(char symbol);
int isOperator(char symbol);
int prec(char symbol);
int top = -1;
char stack[MAX];

main()
{
  char infix[20], prefix[20], temp;
  printf("Enter the infix operation: ");
  gets(infix);
  intoprefix(infix, prefix);
  reverse(prefix);
  puts(prefix);
}

void intoprefix(char infix[20], char prefix[20])
{
  int i, j = 0;
  char symbol;
  stack[++top] = '#';
  reverse(infix);
```

```
for (i=0; i<strlen(infix); i++)
{
    symbol = infix[i];
    if (isOperator(symbol)==0)
    {
        prefix[j] = symbol;
        j++;
    }
    else
    {
        if(symbol == ')')
        {
            push(symbol);
        }
        else if (symbol == '(')
        {
            while (stack[top] != ')')
            {
                prefix[j] = pop();
                j++;
            }
            pop();
        }
        else
        {
            if(precd(stack[top]) <= precd(symbol))
            {
                push(symbol);
            }
            else
            {
                while(precd(stack[top]) >= precd(symbol))
                {
                    prefix[j] = pop();
                    j++;
```

```
            push(symbol);
            }
        }
    }
}
while(stack[top]!='#')
{
    prefix[j]=pop();
    j++;
}
prefix[j]='\0';
}
void reverse(char array[30])
{
    int i,j;
    char temp[100];
    for(i=strlen(array)-1,j=0;i+1!=0;--i,++j)
    {
        temp[j]=array[i];
    }
    temp[j]='\0';
    strcpy(array,temp);
}
char pop()
{
char a;
a=stack[top];
top--;
return a;
}
```

```
void push(char symbol)
{
    top++;
    stack[top] = symbol;
}
int pred (char symbol)
{
  switch (symbol)
  {
     case '+':
     case '-':
          return 2;
          break;
     case '*':
     case '/':
          return 4;
          break;
      case '$':
      case '^':
          return 6;
          break;
       case '#':
       case '(':
       case ')':
            return 1;
            break;
        }
  }
  int isOperator (char symbol)
  { switch (symbol)
     {
       case '+':
       case '-':
       case '*':
       case '/':
```

```
case '^':
case '$':
case '&':
case '(':
case ')':
        return 1;
        break;
default:
        return 0;

}
}
```