```
//InfixToPostfix


#include <stdio.h>

#include <conio.h>

#include <ctype.h>

#include <string.h>

#define MAX 100

char st[MAX];

int top = -1;

void infixtopostfix(char source[], char target[]);

int getpriority(char);

void push(char st[], char);

char pop(char st[]);

int main()

{

   char infix[100], postfix[100];

   printf("Enter any infix expression\n");

   gets(infix);

   strcpy(postfix, "");

   infixtopostfix(infix, postfix);

   printf("The corresponding postfix expression is:\n");

   puts(postfix);

   return 0;

}
```

```
int getpriority(char op)
{
   if (op == '/' || op == '*' || op == '%')
      return 1;
   else if (op == '+' || op == '-')
      return 0;
}
void push(char st[], char val)
{
   if (top == MAX - 1)
      printf("Stack overflow\n");
   else
   {
      top++;
      st[top] = val;
   }
}
char pop(char st[])
{
   char val = ' ';
   if (top == -1)
   {
      printf("Stack Underflow\n");
   }
```

```
    else
    {
        val = st[top];
        top--;
    }
    return val;
}

void infixtopostfix(char source[], char target[])
{
    int i = 0, j = 0;
    char temp;
    strcpy(target, "");
    while (source[i] != '\0')
    {
        if (source[i] == '('
        {
            push(st, source[i]);
        i++;
}
else if (source[i] == ')')
{
    while ((top != -1) && (st[top] != '('))
    {
```

```
    target[j] = pop(st);
    j++;
}
if (top == -1)
{
    printf("\n Incorrect Expression");
    exit(1);
}
    temp = pop(st);
    i++;
}
else if (isdigit(source[i]) || isalpha(source[i]))
{
    target[j] = source[i];
    j++;
    i++;
}
    else if (source[i] == '+' || source[i] == '-' || source[i] == '*' || source[i] == '/' || source[i] ==
'^')
    {
      while ((top != -1) && (st[top] != '(') && (getpriority(st[top]) > getpriority(source[i])))
      {
        target[j] = pop(st);
        j++;
```

```
    }
    push(st, source[i]);
    i++;
}
else
{
    printf("\n Incorrect Element in Expression ");
    exit(1);
    }
  }
  while ((top != -1) && (st[top] != '('))
  {
    target[j] = pop(st);
    j++;
  }
  target[j] = '\0';
}
```

```
/tmp/9yY5kAsMt5.o
Enter any infix expression
(a+b*c)
The corresponding postfix expression is:
abc*+
```