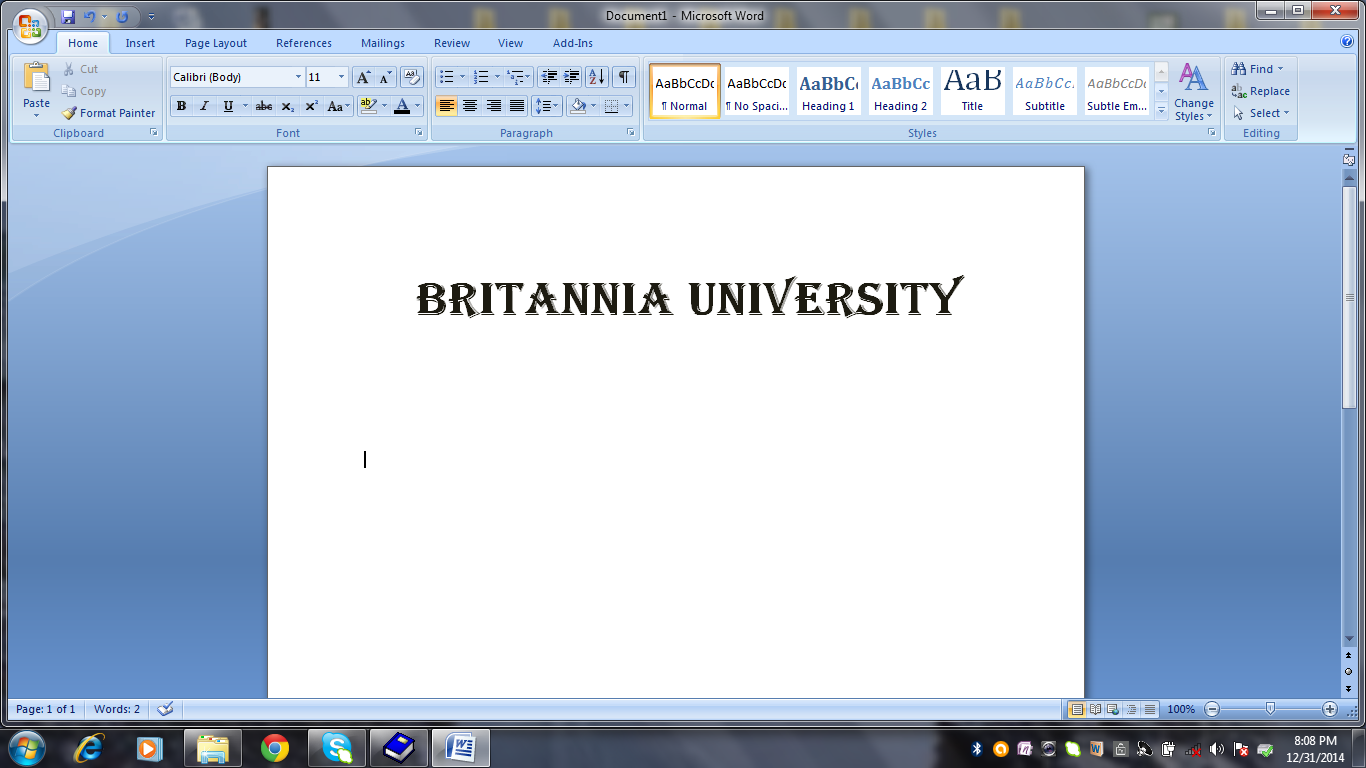


****

**Course title:** Compiler Design.

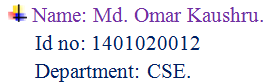
**Assignment on:**

1. A C program to implement context free grammar and check the validity of given expression.
2. A c program to convert infix expression into postfix expression.
3. A c program to convert infix expression into prefix expression.



Date of Submission: 22-08-2016

Submitted by:

****

* A C program to implement context free grammar and check the validity of given expression.

# include <stdio.h>

# include <string.h>

**char** grammer**[**3**][**10**]={"SS+","SS\*"** **,"id"};**

**char** stack**[**100**];**

**int** main**()**

**{**

**char** a**[**100**],**temp1**[**10**],**temp2**[**10**];**

**int** length**,**i**,**j**,**x**,**flag**;**

**while(**gets**(**a**))**

**{**

printf**("\n");**

flag**=**1**;**

**int** length**=**strlen**(**a**);**

**for(**i**=**0**;** i**<**length**;** i**++)**

**{**

x**=**0**;**

**if((**a**[**i**]>=**65 **&&** a**[**i**]<=**90**)** **||** **(**a**[**i**]>=**97 **&&** a**[**i**]<=**122**))**

**{**

stack**[**i**]=**'S'**;**

stack**[**i**+**1**]=**'\0'**;**

**}**

**else** **if(**a**[**i**]==**'+' **&&** i**>**1**)** *///i>1 this condition to check a\*aa+ such case*

**{**

stack**[**i**]=**'+'**;**

stack**[**i**+**1**]=**'\0'**;**

**}**

**else** **if(**a**[**i**]==**'\*' **&&** i**>**1**)**

**{**

stack**[**i**]=**'\*'**;**

stack**[**i**+**1**]=**'\0'**;**

**}**

**else**

**{**

flag**=**0**;**

**break;**

**}**

puts**(**stack**);**

**}**

**while(**flag**!=**0**)**

**{**

j**=**0**;**

length**=**strlen**(**stack**);**

**int** lg0**=**strlen**(**grammer**[**0**]);**

**int** lg1**=**strlen**(**grammer**[**1**]);**

**for(**i**=**0**;** i**<**lg0**;** i**++)**

**{**

temp1**[**i**]=**stack**[**i**];**

**}**

temp1**[**i**]=**'\0'**;**

**for(**i**=**0**;** i**<**lg1**;** i**++)**

**{**

temp2**[**i**]=**stack**[**i**];**

**}**

temp2**[**i**]=**'\0'**;**

**if(**strcmp**(**grammer**[**0**],**temp1**)==**0 **||** strcmp**(**grammer**[**1**],**temp1**)==**0**)**

**{**

j**=**1**;**

stack**[**0**]=**'S'**;**

**for(**i**=**lg0**;** i**<**length**;** i**++)**

**{**

stack**[**j**]=**stack**[**i**];**

j**++;**

**}**

stack**[**j**]=**'\0'**;**

puts**(**stack**);**

**}**

**else** **if(**strcmp**(**grammer**[**0**],**temp2**)==**0 **||** strcmp**(**grammer**[**1**],**temp2**)==**0**)**

**{**

j**=**1**;**

stack**[**0**]=**'S'**;**

**for(**i**=**lg1**;** i**<**length**;** i**++)**

**{**

stack**[**j**]=**stack**[**i**];**

j**++;**

**}**

stack**[**j**]=**'\0'**;**

puts**(**stack**);**

**}**

**else**

**{**

**for(**i**=**0**;** i**<**length**;** i**++)**

**{**

**if(**stack**[**i**]==**'+' **||** stack**[**i**]==**'\*'**)**

**{**

x**=**i**;**

**break;**

**}**

**}**

stack**[**0**]=**'S'**;**

j**=**1**;**

**for(**i**=(**x**-**1**);** i**<**length**;** i**++)**

**{**

stack**[**j**]=**stack**[**i**];**

j**++;**

**}**

stack**[**j**]=**'\0'**;**

puts**(**stack**);**

**}**

**if(**length**==**1**)**

**{**

flag**=**1**;**

**break;**

**}**

**}**

**if(**flag**==**1**)**

**{**

printf**("\n the expression : ");**

puts**(**a**);**

printf**("is valid.\n");**

**}**

**else** **if(**flag**==**0**)**

**{**

printf**("\n the expression : ");**

puts**(**a**);**

printf**("is not a valid expression.\n");**

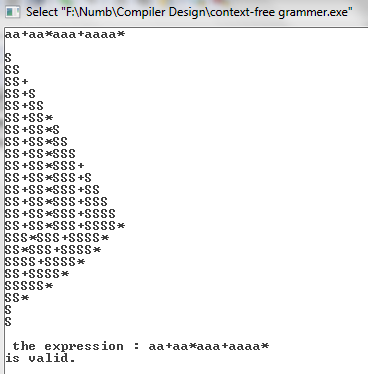
**}**

**}**

**return** 0**;**

**}**

**Sample input/output-1:**



* A C program to convert an infix expression into postfix expression.

#include <stdio.h>

#include <string.h>

#include <ctype.h>

**char** oStack**[**100**],**Postfix**[**100**];**

**int** top**=-**1**;**

**void** push**(char** a**)**

**{**

oStack**[++**top**]=**a**;**

**}**

**int** isOperator**(char** a**)**

**{**

**switch(**a**)**

**{**

**case** '+'**:**

**case** '-'**:**

**case** '\*'**:**

**case** '/'**:**

**case** '^'**:**

**return** 0**;**

**}**

**}**

**char** pop**()**

**{**

**return** **(**oStack**[**top**--]);**

**}**

**int** operatorPrecedence**(char** a**)**

**{**

**switch** **(**a**)**

**{**

**case** '('**:**

**return** 1**;**

**case** '-'**:**

**case** '+'**:**

**return** 2**;**

**case** '\*'**:**

**case** '/'**:**

**return** 3**;**

**case** '^'**:**

**return** 4**;**

**}**

**}**

**int** main**()**

**{**

**char** infixExpr**[**100**];**

**int** i**,**t**,**j**,**k**,**length**,**f**;**

printf**("Enter an infix expression: ");**

gets**(**infixExpr**);**

j**=**0**,**k**=**0**;**

length**=**strlen**(**infixExpr**);**

**for(**i**=**0**;** i**<**length**;** i**++)**

**{**

t**=**0**;**

**if(**infixExpr**[**i**]==**'('**)**

push**(**infixExpr**[**i**]);**

**else** **if** **(**isalnum**(**infixExpr**[**i**]))***///This function checks whether the passed character is alphanumeric*

Postfix**[**k**++]** **=** infixExpr**[**i**];**

**else** **if** **(**infixExpr**[**i**]** **==** ')'**)**

**{**

**while** **(**oStack**[**top**]** **!=** '('**)**

**{**

Postfix**[**k**++]** **=** pop**();**

**}**

pop**();** */// Remove "(" by calling pop to decrease the top position by 1*

**}**

**else** **if(**isOperator**(**infixExpr**[**i**])==**0**)**

**{**

t**=**operatorPrecedence**(**infixExpr**[**i**]);**

**if(**t**!=**4**)***/// because the operator "^" is evaluated right to left so we just push it.*

**while** **(**operatorPrecedence**(**oStack**[**top**])>=**operatorPrecedence**(**infixExpr**[**i**]))**

Postfix**[**k**++]=**pop**();**

push**(**infixExpr**[**i**]);**

**}**

**}**

**while** **(**oStack**[**top**]** **!=** '\0'**)** */// Pop from oStack till it becomes empty*

Postfix**[**k**++]** **=** pop**();**

Postfix**[**k**]** **=** '\0'**;**

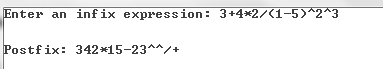
printf**("\n\nPostfix: ");**

puts**(**Postfix**);**

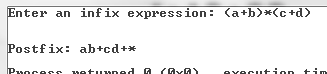
**return** 0**;**

**}**

**Sample input/output:**



**Sample input/output:**



* A C program to convert infix expression into prefix expression.

#include <iostream>

#include <stdio.h>

#include <ctype.h>

#include <string.h>

**using** **namespace** std**;**

**char** oStack**[**100**];**

**int** top**=-**1**;**

**int** isOperator**(char** a**)**

**{**

**switch(**a**)**

**{**

**case** '+'**:**

**case** '-'**:**

**case** '\*'**:**

**case** '/'**:**

**case** '^'**:**

**return** 0**;**

**}**

**}**

**int** operatorPrecedence**(char** a**)**

**{**

**switch(**a**)**

**{**

**case** ')'**:**

**case** '('**:**

**return** 1**;**

**case** '+'**:**

**case** '-'**:**

**return** 2**;**

**case** '\*'**:**

**case** '/'**:**

**return** 3**;**

**case** '^'**:**

**return** 4**;**

**}**

**}**

**void** push**(char** c**)**

**{**

oStack**[++**top**]=**c**;**

**}**

**char** pop**()**

**{**

**return** oStack**[**top**--];**

**}**

**int** main**()**

**{**

**char** infixExpr**[**100**],**Prefix**[**100**];**

**int** i**,**j**,**k**=**0**,**length**,**t**;**

printf**("Enter an infix infixExpression: ");**

gets**(**infixExpr**);**

strrev**(**infixExpr**);**

**for** **(**i**=**0**;** infixExpr**[**i**]!=**'\0'**;** i**++)**

**{**

**if(**infixExpr**[**i**]==**')'**)**

push**(**infixExpr**[**i**]);**

**else** **if(**infixExpr**[**i**]==**'('**)**

**{**

**while(**oStack**[**top**]!=**')'**)**

Prefix**[**k**++]=** pop**();**

pop**();**

**}**

**if(**isOperator**(**infixExpr**[**i**])==**0**)**

**{**

t**=**operatorPrecedence**(**infixExpr**[**i**]);**

**if(**operatorPrecedence**(**oStack**[**top**])<=**operatorPrecedence**(**infixExpr**[**i**]))**

push**(**infixExpr**[**i**]);**

**else**

**{**

**if(**t**!=**4**)**

**while(**operatorPrecedence**(**oStack**[**top**])>**operatorPrecedence**(**infixExpr**[**i**]))**

Prefix**[**k**++]=** pop**();**

push**(**infixExpr**[**i**]);**

**}**

**}**

**else** **if(**isalnum**(**infixExpr**[**i**]))**

Prefix**[**k**++]=**infixExpr**[**i**];**

**}**

**while** **(**oStack**[**top**]** **!=** '\0'**)** */// Pop from oStack till empty*

Prefix**[**k**++]** **=** pop**();**

Prefix**[**k**]** **=** '\0'**;**

strrev**(**Prefix**);**

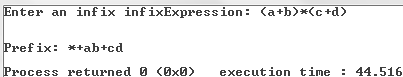
printf**("\n\nPrefix: ");**

puts**(**Prefix**);**

**return** 0**;**

**}**

**Sample input/output:**



**Sample input/output:**

