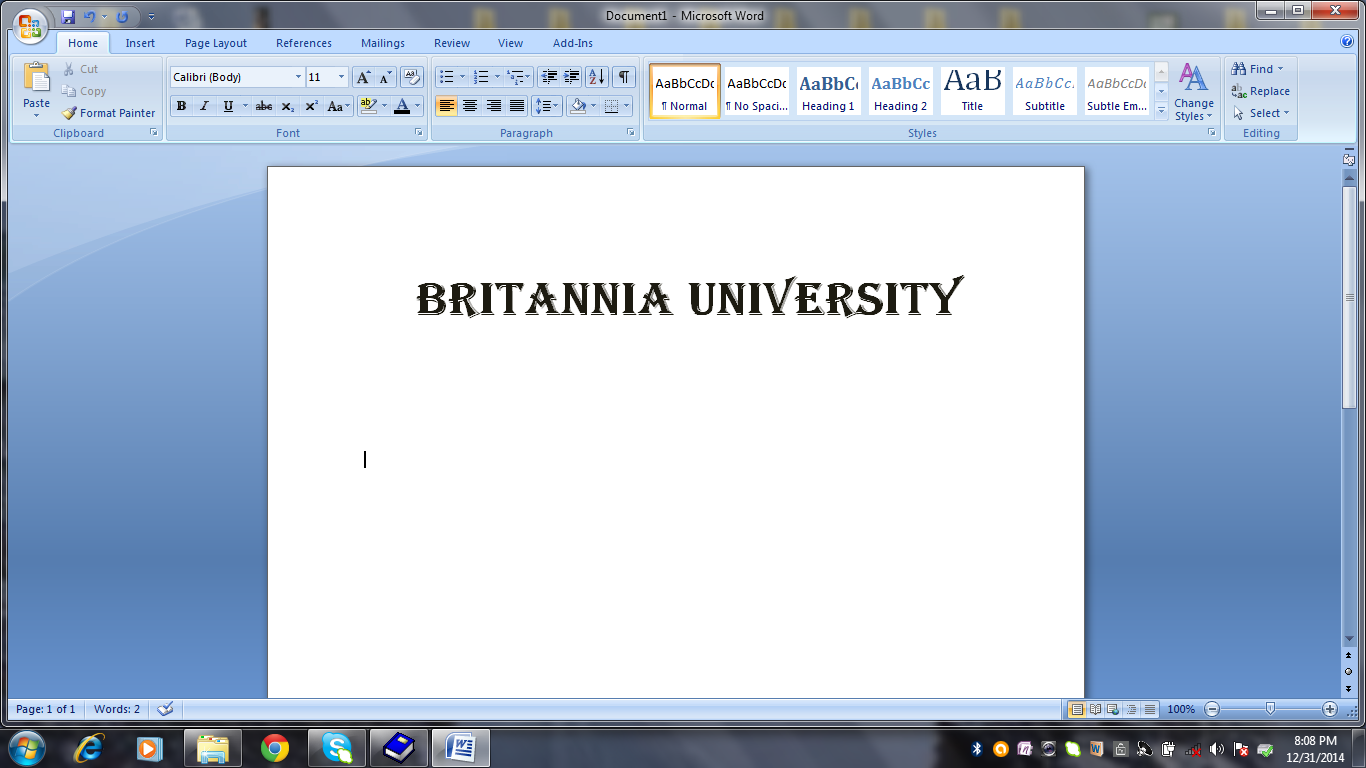


****

**Course title:** Compiler Design.

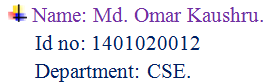
**Assignment on:**

1. A C program to convert postfix expression to prefix expression.
2. A C program to convert postfix expression to infix expression.
3. A c program to convert prefix expression into postfix expression.
4. A c program to convert prefix expression into infix expression.



Date of Submission: 31-08-2016

Submitted by:

****

* A C program to convert postfix expression to prefix expression.

#include <iostream>

#include <cstring>

#include <stack>

#include <algorithm>

#define flag '#'

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

**bool** isOperator**(char** c**)**

**{**

**if(**c**==**'+' **||** c**==**'-' **||** c**==**'\*' **||** c**==**'/' **||** c**==**'^' **||** c**==**'$'**)**

**return** **true;**

**else**

**return** **false;**

**}**

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

**{**

stack**<char>** stk**;**

**char** postfix**[**30**],** prefix**[**30**];**

**int** j**=**0**,**len**;**

cout**<<"Input a postfix expression: ";**

cin**>>**postfix**;**

len **=** strlen**(**postfix**);**

**for(int** i**=**len**-**1**;**i**>=**0**;**i**--)**

**{**

**if(**isOperator**(**postfix**[**i**]))**

stk**.**push**(**postfix**[**i**]);**

**else**

**{**

prefix**[**j**++]** **=** postfix**[**i**];**

**while(!**stk**.**empty**()** **&&** stk**.**top**()==**flag**)**

**{**

stk**.**pop**();**

prefix**[**j**++]** **=** stk**.**top**();**

stk**.**pop**();**

**}**

stk**.**push**(**flag**);**

**}**

**}**

prefix**[**j**]** **=** 0**;**

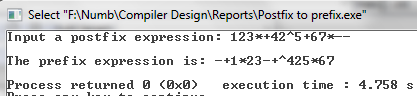
reverse**(**prefix**,** prefix **+** len**);**

cout**<<"The prefix expression is: "<<**prefix**;**

**return** 0**;**

**}**

**Sample input/output:**



* A C program to convert postfix expression to prefix expression.

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#include <iostream>

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

**char** oStack**[**100**],**inFix**[**100**],**opStack**[**100**];**

**int** top**=-**1**,**tops**=-**1**;**

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

**{**

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

**{**

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

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

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

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

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

**return** 0**;**

**}**

**}**

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

**{**

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

**}**

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

**{**

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

**}**

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

**{**

**char** postFix**[**100**],**temp**[**100**];**

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

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

gets**(**postFix**);**

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

length**=**strlen**(**postFix**);**

**for** **(**i**=**length**-**1**;** i**>=**0**;** i**--)**

**{**

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

**}**

i**=**0**;**

**while(**oStack**[**top**]!=**'\0'**)**

**{**

**if(**isalnum**(**oStack**[**top**]))**

inFix**[**i**]=**pop**();**

**else** **if(**isOperator**(**oStack**[**top**])==**0**)**

**{**

strcpy**(**temp**,**inFix**);**

t**=**strlen**(**inFix**);**

**for(**j**=**t**-**1**;** j**>=**0**;** j**--)**

**{**

**if(**isOperator**(**inFix**[**j**])!=**0 **&&** isOperator**(**inFix**[**j**-**1**])!=**0**)**

**{**

**for(**k**=**j**+**1**;** k**<=**t**;** k**++)**

**{**

inFix**[**k**]=**temp**[**k**-**1**];**

**}**

inFix**[**j**]=**pop**();**

**break;**

**}**

**}**

**}**

i**++;**

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

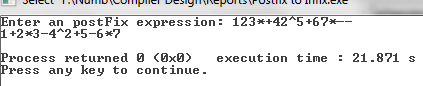
**}**

puts**(**inFix**);**

**return** 0**;**

**}**

**Sample input/output:**



* A c program to convert prefix expression into postfix expression.

#include<iostream>

#include<string.h>

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

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

**char** stack**[**100**][**100**],**temp**,**del**;**

**void** push**(char** ch**[**100**])**

**{**

strcpy**(**stack**[++**top**],**ch**);**

**}**

**void** pop**(char** **\***res**)**

**{**

strcpy**(**res**,** stack**[**top**--]);**

**}**

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

**{**

**char** top\_val\_stack**[**50**],**corr\_top\_val\_stack**[**50**],**prefix**[**100**],\***exp**,**z**[**50**];**

**int** len**,**length**,**length\_of\_exp**,**i**;**

cout**<<"enter a prefix expression:";**

cin**>>**prefix**;**

exp**=**strrev**(**prefix**);**

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

**{**

**if(**exp**[**i**]==**'+'**||**exp**[**i**]==**'-'**||**exp**[**i**]==**'\*'**||**exp**[**i**]==**'/'**||**exp**[**i**]==**'^'**)**

**{**

*/\*pushing a string as a one character\*/*

pop**(**top\_val\_stack**);**

pop**(**corr\_top\_val\_stack**);**

len**=**strlen**(**top\_val\_stack**);**

top\_val\_stack**[**len**++]=**exp**[**i**];**

top\_val\_stack**[**len**]=**'\0'**;**

strcat**(**top\_val\_stack**,**corr\_top\_val\_stack**);**

push**(**top\_val\_stack**);**

**}**

**else**

**{**

z**[**0**]=**exp**[**i**];**

z**[**1**]=**'\0'**;**

push**(**z**);**

**}**

**}**

length**=**strlen**(**top\_val\_stack**);**

length\_of\_exp**=**strlen**(**exp**);**

**if(**length**==**length\_of\_exp**)**

**{**

cout**<<"The infix expression of given expression:";**

cout**<<**top\_val\_stack**<<**endl**;**

**}**

**else**

cout**<<"invalid expression";**

**return** 0**;**

**}**

**Sample input/output:**



* A c program to convert prefix expression into infix expression

#include<iostream>

#include<string.h>

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

**char** postfix**(char** exp1**[**100**]);**

**int** top**=-**1**,**top1**=-**1**;**

**char** stack**[**100**][**100**],**stack1**[**100**],**temp**,**del**;**

**void** push**(char** ch**[**100**])**

**{**

strcpy**(**stack**[++**top**],**ch**);**

**}**

**void** push1**(char** ch**)**

**{**

stack1**[++**top1**]=**ch**;**

**}**

**int** oprt\_priority**(char** op**)**

**{**

**if(**op **==** '+' **||** op **==** '-'**)**

**return(**1**);**

**if(**op **==** '\*'**)**

**return(**2**);**

**if(**op **==** '/'**)**

**return(**3**);**

**if(**op **==** '^'**)**

**return(**4**);**

**}**

**void** pop**(char** **\***res**)**

**{**

strcpy**(**res**,** stack**[**top**--]);**

**}**

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

**{**

**return** stack1**[**top1**--];**

**}**

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

**{**

**char** top\_val\_stack**[**50**],**corr\_top\_val\_stack**[**50**],**prefix**[**100**],\***exp**,**z**[**50**],**post**[**100**];**

**int** len**,**length**,**length\_of\_exp**,**i**,**p**;**

cout**<<"enter a prefix expression:";**

cin**>>**prefix**;**

exp**=**strrev**(**prefix**);**

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

**{**

**if(**exp**[**i**]==**'+'**||**exp**[**i**]==**'-'**||**exp**[**i**]==**'\*'**||**exp**[**i**]==**'/'**||**exp**[**i**]==**'^'**)**

**{**

*/\*pushing a string as a one character\*/*

pop**(**top\_val\_stack**);**

pop**(**corr\_top\_val\_stack**);**

len**=**strlen**(**top\_val\_stack**);**

top\_val\_stack**[**len**++]=**exp**[**i**];**

top\_val\_stack**[**len**]=**'\0'**;**

strcat**(**top\_val\_stack**,**corr\_top\_val\_stack**);**

push**(**top\_val\_stack**);**

**}**

**else**

**{**

z**[**0**]=**exp**[**i**];**

z**[**1**]=**'\0'**;**

push**(**z**);**

**}**

**}**

length**=**strlen**(**top\_val\_stack**);**

length\_of\_exp**=**strlen**(**exp**);**

**if(**length**==**length\_of\_exp**)**

**{**

postfix**(**top\_val\_stack**);**

**}**

**else**

**{**

cout**<<"invalid expression";**

**}**

**return** 0**;**

**}**

**char** postfix**(char** exp1**[**100**])**

**{**

**int** p**=**0**;**

**char** post**[**100**];**

**for(int** i**=**0**;**exp1**[**i**]!=**'\0'**;**i**++)**

**{**

**if((**exp1**[**i**]>=**65 **&&** exp1**[**i**]<=**90**)||(**exp1**[**i**]>=**97 **&&** exp1**[**i**]<=**122**)||**isalnum**(**exp1**[**i**])||(**exp1**[**i**]>=**'0' **&&** exp1**[**i**]<=**'9'**))**

**{**

post**[**p**]=**exp1**[**i**];**

p**++;**

**}**

**else** **if(**exp1**[**i**]==**'+'**||**exp1**[**i**]==**'-'**||**exp1**[**i**]==**'\*'**||**exp1**[**i**]==**'/'**||**exp1**[**i**]==**'^'**)**

**{**

**if(**stack1**[**top1**]!=**'\0'**&&**stack1**[**top1**]!=**'('**)**

**{**

**int** x**=**oprt\_priority**(**exp1**[**i**]);**

**int** z**=**oprt\_priority**(**stack1**[**top1**]);**

**if(**x**>**z**)**

**{**

push1**(**exp1**[**i**]);**

**}**

**else** **if(**x**<=**z**)**

**{**

**while(((**oprt\_priority**(**exp1**[**i**]))<=(**oprt\_priority**(**stack1**[**top1**])))&&(**stack1**[**top1**]!=**'('**))**

**{**

post**[**p**]=**pop**();**

p**++;**

**}**

push1**(**exp1**[**i**]);**

**}**

**}**

**else**

**{**

push1**(**exp1**[**i**]);**

**}**

**}**

**else** **if(**exp1**[**i**]==**'('**)**

**{**

push1**(**exp1**[**i**]);**

**}**

**else** **if(**exp1**[**i**]==**')'**)**

**{**

**while(**stack1**[**top1**]!=**'('**)**

**{**

post**[**p**]=**pop**();**

p**++;**

**}**

**if(**exp1**[**i**+**1**]==**')'**)**

**{**

top1**--;**

**}**

**else** **if(**exp1**[**i**+**1**]!=**')'**)**

**{**

**if(**stack1**[**top1**]!=**'('**)**

**{**

post**[**p**]=**pop**();**

p**++;**

**}**

**else**

top1**--;**

**}**

**}**

**}**

**while(**stack1**[**top1**]!=**'\0'**)**

**{**

**if(**stack1**[**top1**]!=**'('**)**

**{**

post**[**p**]=**pop**();**

p**++;**

**}**

**else**

top1**--;**

**}**

post**[**p**]=**'\0'**;**

cout**<<"The postfix expression of given expression:"<<**post**;**

**}**

**Sample input/output:**

