**Assignment C25**

/\*Implement C++ program for expression conversion as infix to postfix and its evaluation using stack based on given conditions

i. Operands and operator, both must be single character.

ii. Input Postfix expression must be in a desired format.

iii. Only '+', '-', '\*' and '/ ' operators are expected.\*/

===========================================================================

**#include** <iostream>

**#include**<cstdio>

**#include**<cstdlib>

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

**#define** SIZE 50 /\* Size of Stack \*/

**char** s[SIZE];

**int** top=-1; /\* Global declarations \*/

**void** **push**(**char** elem)

{ /\* Function for PUSH operation \*/

s[++top]=elem;

}

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

{ /\* Function for POP operation \*/

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

}

**int** **pr**(**char** elem)

{ /\* Function for precedence \*/

**switch**(elem)

{

**case** '#': **return** 0;

**case** '(': **return** 1;

**case** '+':

**case** '-': **return** 2;

**case** '\*':

**case** '/': **return** 3;

}

}

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

{

**char** infx[50],postfx[50],ch,elem;

**int** i=0,k=0;

cout<<"\nEnter Infix Expression: ";

cin>>infx;

push('#'); //# represent end of input expression

**while**( (ch=infx[i++]) != '\0')

{

**if**( ch == '(')

push(ch);

**else**

**if**(isalnum(ch))

postfx[k++]=ch;

**else**

**if**( ch == ')')

{

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

postfx[k++]=pop();

elem=pop(); /\* Remove ( \*/

}

**else**

{ /\* Operator \*/

**while**( pr(s[top]) >= pr(ch) )

postfx[k++]=pop();

push(ch);

}

}

**while**( s[top] != '#') /\* Pop from stack till empty \*/

postfx[k++]=pop();

postfx[k]='\0'; /\* Make pofx as valid string \*/

cout<<"\nPostfix Expression:\n"<<postfx;

**return** 0;

}

===========================================================================

Enter Infix Expression: (A+B\*C-D)/(E\*F)

Postfix Expression:

ABC\*+D-EF\*/