Shrikrushna S. Zirape
21286.

Problem Statement :-

impliment C++ program for expression conversion as infix to postfix and it's evaluation using stack based on given condition.

1. operands and operator, both must be single characters.
2. Input postfix, expression must be a desired format.
3. Only '+', '-', '*' and '/' operator are expected.

Object! -

1. To impliment. Stack. Data structure.
2. Write a function for converting infix expression into prefix postfix expression

Outcome! -

① To impliment infix to postfix conversion program.
② To make the program menu driven.

⋇ S/W and H/W requirement -

operating system 64 bit (fedora 17)
latest Open Source uptoded Eclipse programming framework.

Theory:-

infix expression are readable and solvable by humans. We can easily distinguisth the order of operators, and also can use the parenthesis to solve the part first during solving mathematical expressions. The computer cannot differentiate the operators and parenthesis easily, thed's why postfix Conversion is needed.

example 1.

infix.	$x+y*z$
prefix	$+x*yz$
postfix	$xyz*+$

Algoritthan:-

Infix To Postfix (infix)
Begin.
for each chareter in infix.
string Postfix

1. If chareter is operand then
   Postfix += chareter.
2. else if. chareter is `(` then
   push it to stack.
3. else if chareter is `)` then
   pon & print stack till stack get empty.

4. else if character is operator.
4.1   if stack is empty
       then push it to stack.
4.2   if stack is not empty.
4.2.1   if (Precedence of character > prece(stack-top))
             then push it to stack.
4.2.2.   if (precedence of charcel < prec of stack-top)
             push charcter to stack.
4.2.3.   else.
4.2.3.1.   while stack is not empty, &&
                 precedence (character < prec.(stack-top))
                 pop & print character.
4.2.3.2. push current operator on stack
     end for
     pop & print remaining character.

Time Complexity ($O(n^2)$)

Test Case	Description	exep O/P	Actual O/P	status.
1	1. Enter String 2. Exit. 1 → A * (B+C)*D	ABC+*D*	ABC+*D*	pass
2	1 → A*B+C*D	ABB*CD* +	AB+CD*+	pass

Conclusion :-
     Successfully implimented the prog
For infix to postfix Conversion.