**//infix to prefix**

**//infix to postfix**

**//postfix to prefix**

**//postfix to infix**

//d=(a+b)/c

**->infix to prefix:**

//1)reverse the exp.

//2)read from left to right

//3)skip the gap and tab.

//4)check the char one by one

//5)if the char found is to be digit or an alpha

//then added to the target string. And write from right to left fashun.

//6)if the char if found is closing par then added to the

//stack.

//7)if the char found is to be operator.

//top most element of satck is poped then compair it

//with scaned char

//1) if the periorty of the stack is higher then add to the tar.

//chart

//1) $

//2) \* /

//3)+ -

//2)if the stack operator's periorty is lower or eq

// the char poped is push backed to stack and char scan

// is also added to the stack.

//8)if the char found is to be opening paranthesis then

//all the stack char added to the target.

//9) closing +opening = empty

//10)rest all char is added to the target.

//11)alphabets/characters/numbers are written in target from right to left.

**->infix to postfix**

//1)no reversing.

//2)read char one by one.

//3)skip gaps .

//4)if you found alpha or digit

//then added to target.

//5)if opening par is found then

//added to stack.

//6)if the char is found is to be operator then

//compair it with stack operator

//1)if stack operator prioritor is higer and eq then added to

//targt.

//2)if the stack periorty is lower then stackoperator

//push back to the satck and char scan is also puah back to

//the stack.

//7)if the closing par is found then all the ele.

//of stack is retrived and added to the target(LIFO).

//8) opening and closing par is clash.

//9)all the ele of the stack is added to the target.(lifo)