30/09/2020
DS

Pseudo code

```
// Declaring ~~writing~~ header files //
#include<stdio.h>
# include <stdlib.h>
# include <string.h>

// Start precedence function F ~~[illegible]~~
   prototype declaration //
int F ( Char Symbol) {
  // forcing switch function to operate
     the symbols which are in infix //
  Switch (symbol) {
  case '+' :                      // for '+' and '-'
  case '-' ; return 2;              it is 2 //.
  case '*' ;
   case '/' ; return 4;
   case '^' ;
   case '$' ; return 5;
   case '(' : return 0 ;
   case '#' ; return -1 ;
   default ; return 8 ;
   }
}
```

// Input precedence function G prototype
declaration //
```
int G (char symbol) {
 // taking switch function to operate the
 symbols. //
  Switch (Symbol) {
   case '+' :
   case '-' : return 1;
   case '*' : ~~2~~
   case '/' : return 3;
   case '$' : ~~5~~
   case '^' : return 6;
   case '(' : return 9;
   case ')' : return 0;
   default : return 7;
  }
 }
```

// Declaring the function prototype ~~to~~
for equation which user ~~to~~ should
put //

```
void infix - postfix (char infix[]){

int top, j, i;
// initialize the top,
        for the int datatype //
```

```
char S[30], postfix[30];
    // initialize stack of size 30 for char
       datatype //

    ~~other//~~  top = -1;  // Stack is empty 1st //
      s[++ top] = '#',  // there will be '#' if
                                  stack is empty //
      j = 0;

    // take loop //

        for(i=0; i< strlen(infix); i++){

            symbol = infix[i]; // it point to the
                                 1st ~~oper~~ symbol in the
                                 infix //

            while (F(S[top]) > G(symbol))
              {
                postfix[j] = S[top --];
                j++;
              } // if F(S[top]) is greater than G(symbol)
             then pop that operand or operator
            and place it in the postfix //

            if (F(S[top]) != G(symbol)){
              S[++ top] = Symbol;
            } // push that symbol and ~~or operator~~
              operands for stack //
else
  top - - ;
}

while (S[top] != '#'){

    postfix [j++] = S[top--];
}

-postfix [j] = '\0';
 puts (postfix);   // ~~we~~ print postfix //.
}

int main ()
{
     char   exp[30]; // initializing expression
                          Of size 30 for char
                          datatype //
     printf ("enter a expression : \n");
     gets (exp); // entering the expression and
                              declaring that //
     infix_postfix (exp);
      return 0;        ; // calling function //
}
```