LAB-3

1)
```
# include <stdio.h>
# include <string.h>
# define MAX 20
Void infix to prefix (Char infix[20], Char prefix[20]);
Void reverse(Char array[30]);
Char pop();
Void push (Char Symbol);
int isOperator(Char Symbol);
int pred (Char Symbol);
 int top = - 1;
Char stack [MAX];

main(){
    Char infix [20], prefix [20], temp;
    printf("Enter infix operation:");
     gets (infix);
      infixtoprefix (infix, prefix);
     reverse (prefix);
     Puts ((prefix));
  }
Void infixtoprefix (Char infix[20], Char prefix[20]){
  int i, j = 0;
  Char Symbol;
  Stack [++ top] = '#';
  reverse (infix);
 for (i=0; i<strlen(infix); i++){
      Symbol = infix[i];
 if( isOperator (symbol)==0) {
        Prefix [j] = Symbol;
          j++;
    }
else {
  if (Symbol == ')') {
     push (Symbol);
}
  else if (Symbol == '(') {
     while (Stack[top] != ')') {
          prefix[j] = pop();
            j++;
      }
     pop();
   }
    else {
   if (prcd (stack[top]) <= prcd (Symbol)) {
          push (Symbol);
   }
     else {
    while (prcd (Stack [top]) >= prcd (symbol)) {
          prefix[j] = pop();
            j++;
      }
         Push Symbol;
      }
     }
    }
   }
   While (stack [top] != '#') {
        prefix [j] = pop();
          j++;
      }
      prefix [j] = '\0';
   }
```

```
Void reverse (Char array[30]){

int i, j;
Char  temp [100];
for (i= strlen (array) -1 , j=0 ; i+1 != 0; --i, j++){
        temp [j]= array[ i];
}
  temp [j] ='\0';
  strcpy (array, temp ); // Copying temp //
  }
   Char pop (){
   Char a;
   a = Stack [top];
    top --;
   return a;
   }

   Void push (Char Symbol){
    topp ++;
   Stack [top ] = Symbol;
     }

  int pred (Char Symbol){

     Switch (Symbol){
         Case '+' :
         Case '-' :
                  return 2;
                   break;
           Case '*' :
           Case '/' :
           return 4;
            break;
Case '$' :
Case '^' :
          return 6;
           break;
Case '#' :
 Case '(' :
  Case ')' :
     return 1;
      break;
  }
}

int isOperator(Char symbol){
Switch (Symbol){
   Case '+'
  Case '-'    :
   Case '*'   :
    Case '/'  :
   Case '^'   :
   Case '$'   :
    Case '&'  :
    Case '(' :
    Case ')' :
         return 1;
          break;
         default :
            return 0;
     }
  }
```

2)
```
# include <stdio.h>
# include <math.h>
# include <string.h>
double Compute(Char Symbol, double op1, double op2)
{
  Switch (Symbol)
  {
     Case '+' : return op1+op2;
     Case '-' : return op1-op2;
     Case '*' : return op1*op2;
     Case '/' : return op1/op2;
     Case '$' :
     Case '^' : return pow(op1, op2);
  }
}

Void main()
{
  double s[20];
  double res;
  double op1, op2;
  int top , i;
  Char postfix[20], Symbol;
 printf ("Enter the postfix expression :\n");
 Scanf ("%s", postfix);
top = -1;
for (i=0; i< strlen(postfix); i++)
{
   Symbol = postfix[i];
    if (isdigit(Symbol))
       S[++top] = Symbol - '0';
else {
    op2 = S[top--];
    op1 = S[top--];
    res = Compute(Symbol, op1, op2);
    S[++top] = res;
  }
 }
 res = S[top--];
 printf("Result = %f\n", res);
}
```

3)

```
# include <stdio.h>
 int fact (int n)
 {
     if (n ==0) return 1;
     return n*fact(n-1);
 }
  Void main()
     int n;
   printf("Enter a number \n");
    Scanf("%d", &n);
   printf("factorial of %d is %d \n", n, fact(n));
 }
```

4)

```
# include <stdio.h>
 int compute(int x, int y){
  int i, great, gcd;
 if(x>y){
  great = x;
 }
  else if(y>x){
      great = y;
  }
    for (i=1; i<great, i++){
     if(x%i==0 && y%i==0){
           gcd=i;
    }
  }
    printf("GCD is %d", gcd);
}
 int main()
 {
      int x,y;
    printf ("Enter 2 numbers \n");
    scanf ("%d %d", &x, &y);
    compute (x,y);
 }
```