CSC 382 **Project 1**

Compare Fibonacci (recursion vs. bottom up)

In this project we will compare the computational time taken by a recursive algorithm to determine the Fibonacci number of an integer *n* and the time taken by a bottom-up approach (using a loop) to calculate the Fibonacci number of the same integer *n*.

A Fibonacci number $F(n)$ is determined by the following recurrence function:

$F(0) = 0$; $F(1) = 1$;

$F(n) = F(n-1) + F(n-2)$, for $n \geq 2$

Thus the ***recursive algorithm*** can be written in C++ as

```
 int FiboR ( int n) // array of size n

{     if (n==0 || n==1)

        return (n);

      else return (FiboR (n-1) + FiboR(n-2));

}
```

And the non-***recursive algorithm*** can be written in C++ as

```
int FiboNR ( int n) // array of size n

  {  int F[max];

      F[0]=0; F[1]=1;

     for (int i =2; i <=n; i++)

          { F[n] = F[n-1] + F[n-2];

          }

  return (F[n]);

}
```

While FiboR takes exponential time FiboNR takes *n* steps

As you will find out the time to compute the recursive Fibonacci will take a long time as you are repeating the same call to an specific value several times.

But you could modify the recursive algorithm to make it very efficient as if a value has been calculated you can store it and if the recursion is trying to calculate the value again you do not need to make recursive calls as you can returned this value. We will call this algorithm **MODFibR. You need to write this algorithm.**

Write an algorithm that computes the time (in seconds using *ctime.h* header library file that takes to determine Fibonacci (*n*) using FiboR and the time taken by FiboNR on the same input *n*.

Try to run both routines using different values of n (n={1,5,10,15,20,25,30,35,40,45,50,55,60…)

You final output should look like:

Fibonacci time analysis (recursive vs. non-recursive)

Integer	FiboR (seconds)	**MODFibR**	FiboNR(seconds)	*Fibo-value*
1	XX.XX	XX.XX	XX.XX	1
5	XX.XX	XX.XX	XX.XX	5
..	..	..		
60	XX.XX	XX.XX	XX.XX	XXXXXXXXXX
…				