Homework#1 (15/100 points)
Due: 2/5/2020 11.55pm

Instructions: Submit one .cpp file for this assignment electronically to the Blackboard system (under Assignment->Homework#1). File in other formats will not be accepted.

WARNING AGAINST PLAGIARISM
Assignments are individual tasks. Students who submit copied code will obtain a mark of zero for the assignment and disciplinary steps may be taken. It is also unacceptable to lend your work to other students – do not make your code available to anyone!

Task1 – *n*th Fibonacci [8 points]

1. Write a C++ program and implement the different Fibonacci functions (as discussed in the class) that return the *n*th Fibonacci number. The template for this assignment is available on the blackboard. Here are the pseudocodes for the Fibonacci functions that you need to implement:

```
fib_recursive(n)
    if n == 1
        return 1
    if n == 2
        return 2
    return fib_recursive(n-1) + fib_recursive(n-2)

fib_memoize(n)
    //initialize memoize[] array with initial values of -1
    memoize[1] = 0
    memoize[2] = 1
    if memoize[n] ≠ -1
        return memoize[n]
    else
        memoize[n] = fib_memoize(n-1) + fib_memoize(n-2)
        return memoize[n]

fib_space_optimized(n)
    counter = 2
    a = 0
    b = 1
    if n ==1
        return 0
    while counter ≤ n
        nextFib = a + b
        a = b
        b = nextFib
        counter = counter +1
    return a
```

You need to output the time it takes to run the three functions when the input size is 1,5, 10, 15, 20, 30, 40 and display the output similar to the ones shown below:

```
n           Recursion   Time        Memoize     Time        Space_OptimizedTime
-------------------------------------------------------------------------------
1           0           0.000116    0           0.077619    0           9.7e-05
5           3           0.00023     3           0.00026     3           0.000111
10          34          0.000571    34          0.000246    34          0.000105
15          377         0.004309    377         0.000231    377         0.000128
20          4181        0.054499    4181        0.000273    4181        0.000157
30          514229      4.57405     514229      0.000495    514229      0.000146
40          63245986    497.41      63245986    0.000722    63245986    0.000175
```

2. Try also with *n*=50 and see what happens. Which algorithm do you think is the most effective? We will also discuss this in the class.

Task 2 – Working with Arrays [7 points]

Complete the three functions with the specification below:

`int algo1(vector<int> & myVec)` – returns the first element of the array

`int algo2(vector<int> & myVec)` – returns the sum of all the elements of the array

`void algo3(vector<int> & myVec, vector<pair<int,int>> & myPair)` – create all possible pairs of the element in the array, for example an array {1,2,3} would generate the following pairs {1,2} , {1,3}, {2,1} , {2,3} , {3,1}, {3,2}. The function then stores the generated pairs in the vector `myPair.`

You need to output the time it takes to run the three functions for the three vectors given in the code template and display the output similar to the ones shown below:

```
n           Algo1       Algo2       Algo3
-----------------------------------------------
10          0.000118    0.000184    0.020171
500         7e-05       0.00214     6.4233
10000       0.000327    3964.25     7.75205e-315
```