Assignment/Homework #1
COP-3530, Fall 2016

Rules & Instructions:

- Due date: Friday, September 9, 2016 at 1 p.m. (Eastern Time)
- This assignment has **3 problems**.
- The assignment/homework will be submitted **by email** to abajuelo@fiu.edu
- Your submission must be a ZIP file (not RAR format). **Please name your submission as 1_xxxxxxx.zip, where xxxxxxx is your seven digit Panther ID number)**.
- Please include the following header for each Java program:

```
/*************************************************************************
   Purpose/Description: <a brief description of the program>
   Author's Panther ID:  <your Panther ID number>
   Certification:
     I hereby certify that this work is my own and none of it is the work of
     any other person.
*************************************************************************/
```

- Please indicate in the **subject of your email message** the following information:

COP-3530, SECTION U05, ASSIGNMENT #1

- Please make sure that you do not include any other personal information in your submission (besides the **Panther ID** in the name of the ZIP file and in the headers of your Java files as explained above). For example, no date of birth or name should be found in the document(s) you submit.
- Submissions turned in after the due date and/or which don't meet the established formatting rules will not be accepted.

Problem #1.

Given an array of integers (there may be duplicates).
Implement in **Java** a **recursive** method that returns **all the possible sums** that can be formed by using numbers in the array.

For example: If the input array contains the integers 1, 2, and 3, the all possible sums are 0, 1, 2, 3, 3, 4, 5, and 6. Notice that the sum 3 must appears twice in the output.

Note: The order of the possible sums in the output is irrelevant.

Problem #2.

The input is a square, *N* by *N,* matrix of positive integers. Each individual row is a **strong decreasing sequence** from left to right. Each individual column is a **strong decreasing sequence** from top to bottom.
Implement in **Java** an **O(*N*)** worst-case algorithm that decides if a number *x* is in the matrix and, if x is present, prints how many times.
Note: Please include in the header of your Java program (using comments) the main ideas of your algorithm.

Example of the 4 x 4 matrix:

$$\begin{pmatrix} 26 & 22 & 17 & 10 \\ 19 & 16 & 12 & 7 \\ 12 & 10 & 7 & 4 \\ 5 & 4 & 2 & 1 \end{pmatrix}$$

Example of the output for this matrix and x = 7

```
Number 7 found in the matrix 2 times
```

Problem #3.

The **Lucas sequence** $\{\text{lucas}\}_{i\geq 0}$ is defined recursively as follows:

lucas(0) = 2, lucas(1) = 1 and, lucas(*n*) = lucas(*n*-1) + lucas(*n*-2) for $n \geq 2$.

The numbers in the Lucas sequence are called the Lucas numbers. For example:

{2, 1, 3, 4, 7, 11, 18, 29, 47, 76, 123, 199, . . .}

(a) Implement a **sub-linear** time complexity function in **Java**.

void findLucas(int n)

that prints the ***n*th Lucas number**

Example of the output of your program:

```
9th Lucas number = 76
```

(b) What is the running time complexity of your function? Justify.

Very important!

1. Your Java program should print correctly long Fibonacci numbers, for example:

   ```
   215th Lucas number = 855741617674166096212819925691459689505708239
   ```

2. In the resolution of this problem it is **not allowed** (and is not a good idea) to use the analog of **Binet's Fibonacci number formula** for **Lucas numbers**.