Assignment 1

Data Preparation Techniques (COMP 3400)

Fall 2024

Important notes:

1. You are required to submit your assignment in *one* file in *IPython Notebook* format (due date: Sep 29). Please note that:
   - You may use *Markdown* in your IPython Notebook.
   - How you develop your IPython Notebook is your choice. You may use *jupyter.org* or *Google Colab*, or a locally installed Jupyter platform on your machine.
2. If not instructed otherwise, use solely `NumPy` (imported as `np`) to solve the problems.
3. In this assignment the term "array" means `Python`'s `ndarray`.
4. You are not allowed to use loops, in any of the problems except for Problem 4 part 1 (or you'll get a mark of 0 for that problem).
5. For some of the problems you may have to refer to the `NumPy's` API.
6. Slides covered in this assignment: *2-01* to *2-06*.

Problem 1 (10 pts). Create a 7×7 array consisting of zeros, and then use *only* slicing (along with value assignment) operations to produce and print the following pattern:

$$\begin{matrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 2 & 2 & 2 & 1 & 0 \\
0 & 1 & 2 & 3 & 2 & 1 & 0 \\
0 & 1 & 2 & 2 & 2 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{matrix}$$

Problem 2 (20 pts). Create two 1d arrays of size 5 consisting of random floating point numbers (these could be the readings from two sensor devices for example). Find the indices of the closest pair among *all the possible* pairs where the first element of a given pair comes from the first array and the second element comes from the second array.

Problem 3 (30 pts). Use masking and aggregation to determine how many integers between 10^6 and $10^7 - 1$ exist where either their square is a multiple of 6 or their cube is a multiple of 15 (but not both). Modify the array of integers from the previous step such that all but such numbers are replaced by random 0's and 1's.

Problem 4 (40 pts). Imagine you have 100,000 right triangles where the lengths of the legs are known and the lengths of the hypotenuses are to be computed. You may use random floating point numbers for the lengths of the legs.

1. Without using `NumPy`, write a `Python` program that computes the lengths of all the the hypotenuses.
2. Employ `NumPy` to write a more efficient program for the problem.
3. Measure the speed-up with `%timeit`.