NT Lab 14: Comparing Sorting Algorithms (25 pts)

Objectives: Write code to compare three different sorting algorithms.

Part 1. [20 points]

Use the starter-code **compare_sorts.py** to write a **main()** function that compares the speed of three sorting algorithms. The sorting algorithms are provided as three functions. **Do *not* edit the sorting functions. Only edit the main() function.** Add code to the **main()** function as follows:

- The starter-code imports two modules: **random** and **time**. As you know, the **random** module has functions for generating random numbers. The **time** module provides the function **time.time()** which returns the current time.
- Use the predefined global variable **maxvalue** to create a random list of **maxvalue** integers between 1 and **maxvalue**. Call **time.time()** to get the start time and save it in a variable, then call **merge.sort()** to sort the list, then call **time.time()** again to get the end time. The elapsed time will be the end time minus the start time. Print the elapsed time for **merge_sort()** to sort the list.
- Generate a *new* **random list** using the same **maxvalue** and measure and print the time for the **insertion_sort** to sort the list.
- Generate a *new* **random list** using the same **maxvalue** and measure and print the time for the **bubble_sort** to sort the list.

Part 2. [5 points]

After you have completed Part 1 and it works, run the program several times with these different values of **maxvalue**: 100, 1000, 10000, and 100000. For each value of **maxvalue**, write down the time required for each sorting algorithm, then using a text editor, **hand-type** the times into a **results.txt** file.

Hint: Some of the sorting algorithms take a <u>really long time</u> to sort 100000 numbers. Be patient. Get a snack or something while you wait for them to finish.

Submit: Upload **compare_sorts.py**, **results.txt**, and **readme.txt** to **NT Lab 14** in Canvas**.**