**CSE 5311**

**Project 1** **Project due date:** Sep 28, 2023, 11:59 PM

For this assignment, you will implement

a) Insertion Sort
b) Merge Sort
c) Quick Sort

Here are the assignment requirements:

- Write three programs for each of the sorting algorithms named as **insertionsort.py, mergesort.py and quicksort.py** (or another extension if the code is in a language other than python). **Code may be implemented in any programming language**.
- For the quick sort algorithm, last element of the array will be the pivot.
- Write **a function in insertionsort.py** that **generates a <u>set of 3 numbers</u>** in a list of 20, 100, 2000, and 6000 **sets of random integers between 0 and 99 integers** and saves these in **4 files: arr20.txt, arr100.txt, arr2000.txt, and arr6000.txt.** The contents of the 4 files are in blue. The "**count of sets**" is given to show how many sets are present in each fie. **The numbers in each row in the input files are space separated** as indicated: "22 33 27"
- The same input files (**arr20.txt, arr100.txt, arr2000.txt, and arr6000.txt as shown below**) generated in insertiosort.py will be used for sorting via insertion sort, merge sort, and Quick sort. This will help you observe the different time it takes each of the algorithm to sort the same number of elements in each file.

| count of sets | arr20.txt | | |
|---|---|---|---|
| 1 | 22 | 33 | 27 |
| 2 | 10 | 15 | 14 |
| 3 | 23 | 14 | 17 |
| 4 | 15 | 99 | 35 |
| ... | ... | ... | ... |
| 20 | 45 | 35 | 15 |

| count of sets | arr100.txt | | |
|---|---|---|---|
| 1 | 32 | 35 | 97 |
| 2 | 30 | 40 | 55 |
| 3 | 56 | 33 | 12 |
| 4 | 67 | 12 | 11 |
| ... | ... | ... | ... |
| 100 | 22 | 24 | 23 |

| count of sets | arr2000.txt | | |
|---|---|---|---|
| 1 | 49 | 66 | 33 |
| 2 | 22 | 65 | 67 |
| 3 | 68 | 30 | 11 |
| 4 | 55 | 78 | 88 |
| ... | ... | ... | ... |
| 2000 | 69 | 55 | 26 |

| count of sets | arr6000.txt | | |
|---|---|---|---|
| 1 | 48 | 27 | 13 |
| 2 | 10 | 18 | 78 |
| 3 | 50 | 15 | 10 |
| 4 | 15 | 70 | 16 |
| ... | ... | ... | ... |
| 6000 | 56 | 76 | 20 |

- Read the contents of the 4 files into a data structure (in each algorithm), compute the sum of each row and save the sum in the same data structure as shown below.

| :ount of sets | arr20.txt | | | |
|---|---|---|---|---|
| 1 | 22 | 27 | 33 | 82 |
| 2 | 10 | 14 | 15 | 39 |
| 3 | 14 | 17 | 23 | 54 |
| 4 | 15 | 35 | 99 | 149 |
| ... | ... | ... | ... | |
| 20 | 15 | 35 | 45 | 95 |

| count of sets | arr100.txt | | | |
|---|---|---|---|---|
| 1 | 32 | 35 | 97 | 164 |
| 2 | 30 | 40 | 55 | 125 |
| 3 | 12 | 33 | 56 | 101 |
| 4 | 11 | 12 | 67 | 90 |
| ... | ... | ... | ... | |
| 100 | 22 | 23 | 24 | 69 |

| count of sets | arr2000.txt | | | |
|---|---|---|---|---|
| 1 | 33 | 49 | 66 | 148 |
| 2 | 22 | 65 | 67 | 154 |
| 3 | 11 | 30 | 68 | 109 |
| 4 | 55 | 78 | 88 | 221 |
| ... | ... | ... | ... | |
| 2000 | 26 | 55 | 69 | 150 |

| count of sets | arr6000.txt | | | |
|---|---|---|---|---|
| 1 | 13 | 27 | 48 | 88 |
| 2 | 10 | 18 | 78 | 106 |
| 3 | 10 | 15 | 50 | 75 |
| 4 | 15 | 16 | 70 | 101 |
| ... | ... | ... | ... | |
| 6000 | 20 | 56 | 76 | 152 |

- Sort the rows **by the sum in the last column only** and display **each row** as the last column in **an ascending order** as shown below.

| count of sets | arr20.txt | | | |
|---|---|---|---|---|
| 1 | 10 | 15 | 14 | 39 |
| 2 | 23 | 14 | 17 | 54 |
| 3 | 22 | 33 | 27 | 82 |
| 4 | 45 | 35 | 15 | 95 |
| ... | ... | ... | ... | |
| 20 | 15 | 99 | 35 | 149 |

| count of sets | arr100.txt | | | |
|---|---|---|---|---|
| 1 | 22 | 24 | 23 | 69 |
| 2 | 67 | 12 | 11 | 90 |
| 3 | 56 | 33 | 12 | 101 |
| 4 | 30 | 40 | 55 | 125 |
| ... | ... | ... | ... | |
| 100 | 32 | 35 | 97 | 164 |

| count of sets | arr2000.txt | | | |
|---|---|---|---|---|
| 1 | 68 | 30 | 11 | 109 |
| 2 | 49 | 66 | 33 | 148 |
| 3 | 22 | 65 | 67 | 154 |
| 4 | 69 | 55 | 26 | 150 |
| ... | ... | ... | ... | |
| 2000 | 55 | 78 | 88 | 221 |

| count of sets | arr6000.txt | | | |
|---|---|---|---|---|
| 1 | 50 | 15 | 10 | 75 |
| 2 | 48 | 27 | 13 | 88 |
| 3 | 15 | 70 | 16 | 101 |
| 4 | 10 | 18 | 78 | 106 |
| ... | ... | ... | ... | |
| 6000 | 56 | 76 | 20 | 152 |

- Sort each of the four files using the three algorithms on the last column(sum) -one at a time.

- **Output files** display a list of the sorted row of numbers. Each file should state at its end, the time taken to sort the set of numbers using the selected algorithm. For example, arrIS_O_20.txt should have the 20 sets of numbers, sorted by the last column and in the end, it should state the time taken to sort these using <u>Insertion Sort</u>.
  - **Output files** Insertion sort (**arrIS_O_20.txt, arrIS_O_100.txt, arrIS_O_2000.txt, and arrIS_O_6000.txt**)
  - **Output files** Merge sort (**arrMR_O_20.txt, arrMR_O_100.txt, arrMR_O_2000.txt, and arrMR_O_6000.txt**)
  - **Output files** Quick sort (**arrQK_O_20.txt, arrQK_O_100.txt, arrQK_O_2000.txt, and arrQK_O_6000.txt**)

- Maintain the time taken to sort in the **same unit** for all sorting algorithms for arrays of all sizes.
- Print/display the time taken for your implemented algorithm (In every execution), to sort the input integer along with the sorted array.
- Ensure that your code is readable (indented and commented).
- Submit a README file that explains how to run the code including any environment dependency (language versions), Make Sure your TA can run your code. TAs may ask you to demo your project. **The student absent during the demo will receive a zero**.
- **Project will be completed by and submitted by groups of two students**. Only 1 student will submit the assignment, but both will be present for the demo.
- All files need to be zipped in a zip file before submission that is named as follows:
  - CSE5311-Sectionno-P1-1000XXXXX-1000YYYYY
  - If **Section no is 2 or 9 or 11 or 12** and the Student ids of the two students are 1000XXXXX and 1000YYYYY, the zip file will be named as shown here:
    - CSE5311-02-P1-1000XXXXX-1000YYYYY
    - CSE5311-09-P1-1000XXXXX-1000YYYYY
    - CSE5311-11-P1-1000XXXXX-1000YYYYY
    - CSE5311-12-P1-1000XXXXX-1000YYYYY
- Zip file CSE5311-Sectionno-P1-1000XXXXX-1000YYYYY will contain the following:
  - 3 source code files for three sorting algorithms
  - 4 data files **arr20.txt, arr100.txt, arr2000.txt, and arr6000.txt**
  - **Output files**
    - **Output files** Insertion sort (**arrIS_O_20.txt, arrIS_O_100.txt, arrIS_O_2000.txt, and arrIS_O_6000.txt**)
    - **Output files** Merge sort (**arrMR_O_20.txt, arrMR_O_100.txt, arrMR_O_2000.txt, and arrMR_O_6000.txt**)
    - **Output files** Quick sort (**arrQK_O_20.txt, arrQK_O_100.txt, arrQK_O_2000.txt, and arrQK_O_5000.txt**)
  - **1 report**
- Your **report documents** the following:
  - **List any sites/sources referred**
  - Time complexity of each of the algorithm
  - Your experimental results (**time it takes** for algorithm to sort **4 lists of data structures of varying lengths**).
  - Explain any differences between the experimental and theoretical results.
  - Compare and contrast the results between the three sorting algorithms and time taken to sort the 4 arrays. Explain anomalies if any.
  - Report of 1-2 pages should be enough.
  - Honor code (stated in red in this document)
- Implement your code yourself without copying from anywhere
- **The code will be tested for plagiarism. Any plagiarism will result in zero for all students involved**

**Some rules to follow:**

1. The project is to be completed in a **group of two students** from the same section.
2. Include both names and **IDs** in your document along with a description of **what each member did.**
3. **Handwrite, sign, and date (with date of submission) a copy of the Honor Code (shown below) and share the image as part of your project; a handwritten, signed, and dated (with the date of submission) copy of the Honor Code must be included with every project and exam submission. (Failing to include will cost 20 points)**
4. **Students are required to NOT share their solutions to the project even after the semester is over or even after graduation. However, they can show their projects during their interviews. They are also required to not discuss the solution with others or use anyone else's solution. Any violation of the policy will result in a 0 for this project for all students concerned.**

**HONOR CODE**

I pledge, on my honor, to uphold UT Arlington's tradition of academic integrity, a tradition that values hard work and honest effort in the pursuit of academic excellence.

I promise that I will submit only work that I personally create or that I contribute to group collaborations, and I will appropriately reference any work from other sources. I will follow the highest standards of integrity and uphold the spirit of the Honor Code

I will not participate in any form of cheating/sharing the questions/solutions.