The objective of this task is to study how the theoretical analysis of a variety of sorting algorithms compares with their actual performance. The sorting algorithms you will study are:

- Exchange Sort
- Mergesort
- Quicksort
- Heapsort
- Radix Sort

The major emphasis of this task is on analyzing the performance of the algorithms, NOT on coding the algorithms. Therefore, you can simply use your implementation of these sorting algorithms in your attendance quiz. Please carefully check your implimentation to ensure that it is correct and efficient, otherwise it will influence the analysis result. In this project, most of your time should be spent on designing careful test cases and analyzing your results in order to draw useful conclusions regarding the performance of the various algorithms.

1. Theoretical question (15 marks)
   Assume the n input elements are distinct integers in the range $[1, n]$. For each algorithm, determine what are best, average, and worst-case inputs. You should list these for each algorithm, including a sentence or two of justification for each one. You should answer what you expect to be true based on a theoretical analysis (and you should not refer to experimental results). In the subsequent questions we will compare the experimental results to these theoretical predictions.

2. Data generation and experimental setup (15 marks)
   The choice of test data is up to you (i.e., for each sorting algorithm, which input sizes should be tested, how many different inputs of the same size, which particular inputs of a given size.) Be smart about which experiments to run, i.e., don't run larger or more tests than you need to answer the above questions reasonably well. Also, note that you may need to run your experiments several times in order to get stable measurements. Your experimental setup must be described in terms of the following:
   - What kind of machine did you use?
   - How to get the experimental running time?
   - How many times did you repeat each experiment?
   - What time unit is used?
   - How did you select the inputs?
   - Did you use the same inputs for all sorting algorithms?

3. Analysis of Quicksort (20 marks)
   You will study different strategies for selecting the pivot:
   - Pivot Choice 1: The first element in the array.
   - Pivot Choice 2: A random element in the array.
   - Pivot Choice 3: The median of the first, middle, and last elements in the array.

   You will then have three different versions of Quicksort. Now, you need to compare them with three input cases by graph plot. In each figure, you have three lines for three Quicksort versions. The x-axis is the input size n and y-axis is the time.
   - Graph the best case running time as a function of input size n for all three versions (use the best case input you determined in each case in part 1).

  - Graph the worst case running time as a function of input size n for all three versions (use the worst case input you determined in each case in part 1).
  - Graph the average case running time as a function of input size n for all three versions.

   Analyze each case and give a brief summary.

4. Analysis of five sorting algorithms. (20 marks)
   Compare all the five sorting algorithms with three input cases by graph plot (consider the best version of Quicksort based on your analysis in question 3). In each figure, you have five lines for five sorting algorithms. The x-axis is the input size n and y-axis is the time.
   - Graph the best case running time as a function of input size n for the five sorts (use the best case input you determined in each case in part 1).
   - Graph the worst case running time as a function of input size n for the five sorts (use the worst case input you determined in each case in part 1).
   - Graph the average case running time as a function of input size n for the five sorts.

   Analyze each case and give a brief summary.

5. Compare experimental results with theoretical results. (30 marks)
   To what extent does the best, average and worst case analyses (from class / slides / textbook) of each sort agree with the experimental results? To answer this question, you would need to find a way to compare the experimental results for a sorting algorithm with its theoretical results. One way to compare a time obtained experimentally to the theoretical time complexity of $O(f(n))$ (e.g., $f(n) = n \log n$) would be to divide the experimental time by $f(n)$ and see if you get a horizontal line (after some input size N), for a number of runs with different input sizes n. Because if the experimental time is consistent with the theoretical result, they should be different only on a constant c as shown in the asymptotic analysis.

   For each sort, and for each case (best, average, and worst), determine whether the observed experimental running time is of the same order as the theoretical results. If you found the algorithm didn't conform to the theoretical results, you should try to understand why and provide an explanation. (hint: the theoretical analysis only considers the number of comparisons)

Reference materials:
1. Markdown tutorial: https://www.markdowntutorial.com/
2. Python matplotlib tutorial: https://matplotlib.org/3.2.1/tutorials/introductory/usage.html#sphx-glr-tutorials-introductory-usage-py