# COMP 1001 (Fall 2024): Practice Problem Set 7

Suggested Completion Date: Nov 27, 2024

---

**Important Notes:**

- Practice Problem Sets are not a part of the evaluation for the course.
- After the Suggested Completion Date, a rubric and solution outline will be posted on the course Brightspace shell so you can check your answers.
- Completing the Problem Sets will help you to gain hands-on experience with coding in Python, and will help you apply the concepts we have covered to solve and code solutions for the given problems. This will help you prepare for your tests and exam.
- While coding solutions for the problems given below, keep in mind that on the test/exam you will also be marked on the following:
  - *Efficient solution of the problem.* A given problem can be solved in a number of different ways, but the best solution is the one that is efficient; i.e., the one that uses the right concepts in a very productive way.
  - *Including sufficient descriptive comments in your program.* The person looking at your code should be able to understand how your code is solving the given problem even if the person reading your Python program does not know the Python language. In addition, the reader of your program should be able to understand what each variable represents.
  - *Labelling of input and output.* All input and output should have a descriptive label so that the reader of your program understands what input is expected and what output the program has generated.
  - *Program style* - consistent formatting and indentation of program statements, meaningful variable names (identifiers), and the use of constants (constant identifiers), where appropriate.

  Practicing these rules will build a good foundation for programming.
- This Problem Set is based on Chapter 9 and 10, without the graphics components. Please use only concepts from Chapters 1–6, 9–12 of the textbook.

1. Write a Python program, in a file called `string_merge.py`, that includes the following functions:

   - `mergeRecursive`, a **recursive** function that takes two strings of ASCII-ordered characters and merges them, leaving the characters in ASCII order. Note that the strings may be different lengths.
   - `orderedMerge`, a **non-recursive** function that takes two strings of ASCII-ordered characters and merges them, leaving the characters in ASCII order. Note that the strings may be different lengths.
   - a `main` function that inputs two ordered strings and calls the `mergeRecursive` and `orderedMerge` functions. Your `main` function should print the two input strings and the results of both merge functions.

   **Note:** You **should not** use any of the built-in methods for string concatenation or ordering, but you may use the `len()` function and the slice operator (`:`). You may assume that the input strings are already ordered.

   Sample input/output:

   ```
   Enter the first ordered string: acdrt
   Enter the second ordered string: bdet
   First string: acdrt
   Second string: bdet
   Result of recursive function: abcddertt
   Result of non-recursive function: abcddertt
   ```

   More sample input/output:

   ```
   Enter the first ordered string: DEab
   Enter the second ordered string: BFxz
   First string: DEab
   Second string: BFxz
   Result of recursive function: BDEFabxz
   Result of non-recursive function: BDEFabxz
   ```