ECE 36800 – Data Structures
Programming Assignment 1 – Part 1

Purpose of Assignment 1:
Implement and compare the running time of different sorting algorithms.

Goal of Part 1:
Set up the program for necessary components in preparation for part 2, including reading a word list file, sorting the words using bubble sort, and checking if the word list included are sorted, etc.

What to submit:

1. Create a simple text file with less than 10 word strings. The first line of the file contains the number of strings contained in the file, and each of the remaining lines contains a single string.
2. `sorttest1.cpp`: The test sorting program. Start with the version provided.
3. A word document `proj1p1.docx`: This file should have the following parts:
   - Test result: Run your program with your input text file. Make sure to include the command you used to run the program, any program printout, and the content of the output text file (with the sorted words).
4. Push all your files under the "proj1/part1" directory before the deadline.

Guideline:

- After you read the words from the input file, these words will need to be stored in a string array.
- The `bubblesort`() function sorts the elements in data array in alphabetic order.
- You can directly compare two string variables s1 and s2. If s1<s2, s1 precedes s2 in alphabetic order.
- Type `size_t` is a typedef that's an alias for some unsigned integer type, typically `unsigned int` or `unsigned long`, but possibly even `unsigned long long`. Each Standard C implementation is supposed to choose the unsigned integer that's big enough--but no bigger than needed--to represent the size of the largest possible object on the target platform. For further reading, refer to: https://www.embedded.com/why-size_t-matters/.

Grading Policy:

Part 1 counts for 40% of the overall points in Programming Assignment 1.

Please make sure your program compiles successfully. If not, 20 points will be deducted.

1. **Executability** (5%):
   - Runtime errors: You program must not have runtime errors (e.g., code crash, infinite loop, reading uninitialized memory, accessing the content of a NULL pointer, etc.).

2. **Programming style** (5%):
   - Code efficiency: Code should use the best approach in every case.
   - Readability: Code should be clean, understandable, and well-organized. Please pay special attention to indentation, use of whitespace, variable naming, and organization.
   - Documentation: Code should be well-commented with file header and comments.
3. **Program Specifications/Correctness** (30%):

   Please refer to the Grading Criteria table for details. Specifically, your program should behave correctly, adhere to the instructions, and pass the test program.

file	**item**	**weight (%)**
sorttest1.cpp	Read in word list and store in a string array	5
	Call bubblesort to sort the string array	4
	Write words to the output file	5
	check_ordered function	6
proj1p1.docx	test result	10
	total	**30**