CSE 333 25au Exercise 1

out: Friday, September 26, 2025
due: Monday, September 29, 2025 by **10:00 am**. **No late exercises accepted.**

Goals: Implement and use a function with array parameters; process and print fixed-width integer data.

Description: Write (in C) a function `CopyAndSort` that accepts two arrays of `int64_t` 's and an array length as arguments. You should assume the lengths of the two arrays are the same. The function should iterate through the elements of the first array and use insertion sort to store them into the second array in non-decreasing (i.e., ascending, but there might be duplicates) sorted order. Your code should insert the array elements in the proper place one at a time as they are copied. Do not copy the entire array first and then sort it.

You should assume that the array lengths and subscripts can be stored in variables and passed as parameters of type `int` . When printing array values you should use the correct format for `int64_t` values rather than `printf` codes for built-in C types like `int` , `short` , or `long` (Hint: explore the `stdint.h` and `inttypes.h` libraries). You should decompose the problem into multiple functions if appropriate.

(Note about `int` : in systems code we very often will prefer to use types like `int32_t` with precisely known sizes, but we also want to follow the Google style guide, which says that if all you need is an `int` , you should use `int` . See the Google C++ style guide for more details. The information in there that is not C++-specific also applies to our C code. For this exercise we'll use arrays whose elements have explicitly-sized integer elements, but the array sizes and indices are ordinary integer values.)

Write a `main()` function that exercises your `CopyAndSort` function. When your program runs, it should sort the following array and print out the results:

```
{3, 2, -5, 7, 17, 42, 6, 333, 7, 8, -8, 6}
```

When your program compiles and runs, we should see:

```
bash$ gcc -Wall -g -std=c17 -o ex1 ex1.c
bash$ ./ex1
-8 -5 2 3 6 6 7 7 8 17 42 333
bash$
```

Your code must:

- compile without errors or warnings on CSE Linux machines (lab workstations, attu, or CSE home VM)
- have no crashes, memory leaks, or memory errors on CSE linux machines

- be contained in a single file called `ex1.c`
- be pretty: the formatting, modularization, variable and function names, and so on must make us smile rather than cry (Suggestion: use `cpplint.py --clint` from hw0 to check for potential style problems.)
- be robust: you should think about handling bogus input from the user (if any), and you should handle hard-to-handle cases (if there are any) gracefully
- have a comment at the top of your .c file with your name, and CSE or UW email address

You should submit your exercise using the Gradescope dropbox linked on the course resources web page.
