CS310 Project 2

The purpose of this project is to experiment with sorting algorithms.

Take several mystery sorting algorithms and determine which ones they are by experimentation.

The object file **`mysterysorts.o`** contains six sorting algorithms, which you can use by calling the `mysterySort` function with an array, its length, and an integer between 0 and 5 to indicate which algorithm to use. The function is designed to take an array of numbers and order the elements into ascending order using the particular algorithm. More specifically, the algorithms used are:

Bubble sort
Insertion sort
Selection sort
Heap sort
Merge sort
Quicksort (where first element of the vector is used as the pivot for partitioning.)

Your job as the sorting detective is to figure out which algorithm is implemented by each mystery sort. Below are some techniques you may use.

First, you can interrupt the sort and observe how the values are being moved around the vector. Read the comments of the provided header file that tells you how you can interrupt the sorting operations. You may also want to construct your array with values that are easy to see how values are being moved. The known invariants of each sort algorithm will be important information for identifying the algorithm.

Secondly, you can time the sorting algorithms and get an overall idea of their performances. A way to measure elapsed system time for programs is to use function `gettimeofday()`. If you record the starting and finishing times in the `timeval` structures `start` and `finish`, you can compute the time elapsed in microseconds as demonstrated in the starter file named **`demo.c`** .

To perform this task you will need to complete the **`time_category`** and **`classify`** functions in the file **`detective.c`** . They should take a number for algorithm to select and use one or more calls to `mysterySort` with that algorithm number to determine which algorithm it is using. Do not hard code your conclusions of which algorithm is which, but write code to analyze the algorithm's behavior and determine which algorithm it is. Grading tests will be run with the algorithms being assigned different numbers.

Note that the object file **`mysterysorts.o`** is compiled on the CS department servers under Linux, so it may not be compatible with your home machine.

Submission:

- Submit the file **`detective.c`**.

As always, do not turn in executables or object code, and make sure your submission compiles successfully on the CS servers using the provided makefile. Programs that fail to compile will receive a zero mark (even if it might work perfectly on your home computer.)

Important:

You must include a standard comment block in each of your source files, including your name, section, date and collaboration details.

You should also include detailed collaboration declaration information in your comments. If you worked in pairs in this lab, each of you must include the partner's name in your program comment. Both you and your partner must complete an individual submission in order to earn a grade. If you work by yourself, you must indicate in the program comments that you have worked on your own independently.

You can resubmit as needed—your last submission will be graded. Here is a tentative grading scheme:

	Correct sort	Correct time
Sort 0	5	5
Sort 1	5	5
Sort 2	5	5
Sort 3	5	5
Sort 4	5	5
Sort 5	5	5