You are going to write a generic sort routine in standard C. You do NOT need to make the sort efficient. I recommend implementing bubble sort, selection sort, or insertion sort (whichever you think would be the simplest). However, your sort should be able to sort arrays of any types, when passed the appropriate parameters, including a pointer to a function to perform the "less than" operation for values of the type being sorted.

The main sort function will be similar to the "lsearch" example covered in the lectures that we watched. The parameters should be: (1) a void pointer that points to the array to be sorted; (2) an integer that stores the number of elements in the generic array; (3) an integer that stores the number of bytes of each element in the array; and (4) a pointer to a function that takes two void pointers and returns an int, such that the return value is one if the value pointed to by the first void pointer is less than the value pointer to be the second void pointer, and the return value is zero otherwise. The sort function should not return anything; it should directly modify the array being sorted.

You are also going to write a program that demonstrates the abilities of your search function. To do this, you will create three arrays. One will be a small array of integers (size 10 should be enough). One will be a small array of C-style, null-terminated strings (i.e., an array of pointers to char). One will be a small array of student structures. Before creating this array, you will have to define a student struct, such that each struct stores a student's name (a C-style, null terminated string) and the student's gpa (a double).

To sort the arrays of integers and strings, you will also need to create simple "less than" functions to compare int values and C-string values. The function for comparing strings can rely on "strcmp". You will sort the student array twice, using two different "less than" functions. One should compare students based on their names (this will be similar to the function used for sorting the string array), and the other should compare students based on their gpas.

The point here isn't to write particularly elegant or efficient code. As previously stated, I do not expect the sort to be efficient, and it's OK to put a lot of the code in "main", except for the sort and the "less than" functions. The point here is to make sure you understand the important, low-level details from the lecture's we have been watching and discussing. These include details about pointers including void pointers, how things are stored in memory, function pointers, and how to write generic code in standard C. That being said, I do want you to indent your code nicely and to include appropriate comments. At the very least, include a comment at the top with your name, above each function, above the definition of any structure, and within functions wherever you think code is non-obvious or otherwise interesting.

When you are finished, email me your code to carl.sable@cooper.edu. Only email the code, not your executable or other binary files. I will compile and run the code on my machine. Once I grade it, I don't allow re-submissions; so, if you want to talk to me about your code before submitting it, set up an appointment with me. Of course, you are also welcome to email me questions related to the assignment.