Homework #5 - Sorting

Name:

1 Intro

You've now learned enough C++ that we can start talking about the idea of an **algorithm**. An algorithms is the way you do something. For instance, an algorithm for washing dishes might looks like:

1. Turn on sink
2. Put soap on sponge
3. Pick up dish (bowl, plate, fork, spoon, knife, or cup)
4. Wash with sponge
5. Dry
6. Go back to step 3
7. Stop when there are no more dishes

Actually, an algorithm is not too different from a program. It's a list of instructions that tell you **how** to do something. The difference is that usually an algorithm only talks about how to do one very specific job, usually something mathematical like searching or sorting. Programs often have a lot of **other** work to do like opening and closing files, waiting for users to click on things, etc... and algorithms do not deal with that.

2 Sorting

We talked a little bit about sorting today in class. There are many different ways to sort, and **how** you sort can affect how quickly you can sort. For homework, we'll try to write two simple sorting algorithms, Bubble Sort and Insertion Sort.

For most problems, Bubble Sort is very slow, but there are some times when it is the right choice. For instance, if most of your lists look like this:

```
1 2 3 4 5 6 8 7 9 10 11 12
```

Notice that only 8 and 7 are not in the right order in this list. For lists that are **almost** in the right order already with some things only one or two places away from the correct position, then Bubble Sort is a good choice. We will talk more about this later in the class. Not only does the **algorithm** you use determine how fast you can solve a problem, but sometimes the problem can help you decide which algorithm is the right one to use. Not every kind of data is the same, so there is no **best** algorithm for all situations.

Remember, in algorithms there is **no substitute for understanding the problem you are trying to solve**.

2.1 Bubble Sort

Bubble Sort is perhaps the simplest sorting algorithm. Bubble sort for a list of numbers works like this:

1. Look at the 1st and 2nd numbers in the list
2. If the numbers are not in the right order, flip them
3. Look at the 2nd and 3rd numbers in the list
4. If the numbers are not in the right order, flip them
5. Continue to the end of the list, flipping any out of order number pairs
6. Start again from the beginning of the list
7. If you go through the whole list without flipping any numbers, stop

You can use the code in emptySort.cpp to get started. You just need to write the sorting code in the middle. The code to create and print the array(s) is already written for you in that file.

2.2 Insertion Sort

Below is the pseudo-code for insertion sort. Your job is to turn this into a real C++ program which you can show me in the next class.

```
// Insertion sort pseudo-code
for j = 1 to A.length-1
    key = A[j]
    i = j-1;
    while i >= 0 and A[i] > key
        A[i+1] = A[i]
        i = i - 1
    A[i+1] = key;
```

insertionSortPseudo.cpp

Think about how you could convert this into a real C++ program. How does it work? What happens when this code finds two numbers that are already in order? What about two numbers that are **not** in order? You can start with the C++ code on the next page (emptySort.cpp)

```cpp
#include <iostream>

using namespace std;

int main(){

    int simpleArray[] = {6, 5, 4, 3, 2, 1};

    // Print original array
    cout << "Original array: " << endl;
    for (int i = 0; i < 6; i++){
        cout << simpleArray[i] << " ";
    }
    cout << endl;

    // YOUR CODE HERE

    // Print sorted array
    cout << "Sorted array: " << endl;
    for (int i = 0; i < 6; i++){
        cout << simpleArray[i] << " ";
    }
    cout << endl;

    return 0;
}
```

emptySort.cpp

Your new code should go in the middle, between the two for loops.

2.3 Ok...what about big to small?

Write a C++ program that sorts in reverse. Big numbers first, small numbers last. You can start with the code you wrote for the last question.

2.4 A final challenge

How would you sort words from A-Z? Think about an algorithm to do this. You don't need to write real code, but please write the **pseudocode** for your algorithm here (or on the back of the page if you run out of room):

1.
2.
3.
4.
5.
6.
7.