CS 1428
Fall 2018
Gentry Atkinson
Lab 10

Introduction:

Computer programs often have to deal with long lists of information. You have had some experience with this over the last few weeks. But part of properly maintaining a list of values is creating some way to search and sort those values. **Searching** selects one value from amongst all the others based on some criteria. **Sorting** puts those values in an order determined by some criteria (e.g. alphabetical by name or numerical by price).

There are many, *many* different ways to sort values. Today's lab isn't focused on teaching you the different sorting algorithms but rather to familiarize you with the basic concepts. Feel free to ask or use Google if you want to learn more about specific sorting algorithms. Searching, by contrast is fairly straightforward. We implemented a simple search algorithm during Lab 9 as part of the updateProperty function.

The purpose of today's lab is to familiarize you with the creation and use of sorting algorithms.

Directions:

1- Launch Code::Blocks and start a new file. Name it your_last_name_lab10.cpp. Be sure that you've downloaded lab10_data.txt and that it's in the same folder as your new program.

2- Include the standard header for this lab:

   //Your Name
   //CS1428 Fall 2018
   //Lab 10

3- Include the iostream, time, and fstream standard libraries and declare some functions that we will be using in this lab. Start your main function:

   #include <iostream>
   #include <time.h>
   #include <fstream>

using namespace std;
int main() {

4- Debug the following function and add it to your program:

```
//Swaps the values at indexA and indexB in inputArray
//Save value a into temp, value b into a, then temp into b
void swap (int indexA, int  indexB, int inputArray[])
        int temp = inputArray[indexA];
        inputArray[indexA] = inputArray[indexB];
        inputArray[indexB] == temp;
        Return  0;
```

5- Copy the following function prototype into your code and then implement the pseudo-code to create your first sorting algorithm:

```
void bubbleSort(int toSort[], int arraySize);

//SET endingIndex to arraySize-1
//ENTER OUTER LOOP
        //LOOP THROUGH ALL INDEXES FROM 0 TO endingIndex
                //COMPARE EVERY TWO CONSECUTIVE VALUES (I.E. 0 AND
                1, 1 AND 2, 2 AND 3,…)
                //IF THE VALUE AT THE LARGER INDEX IS SMALLER (I.E
                toSort[3] > toSort[2]) call  swap(smallerIndex, largerIndex,
                inputArray) TO SWAP THE VALUES
        //EXIT INNER LOOP
        //DECREMENT endingIndex
//EXIT OUTER LOOP WHEN endingIndex EQUALS 0
//RETURN
```

After this function runs it should leave the array **toSort** in order from smallest to biggest.

6- Copy the following function prototype into your code:

```
void mySort(int toSort[], int arraySize);
```

Now use your creativity and knowledge of computer science to create a new sorting function. If you need help consider the following strategies:

1. Search the array for the smallest value. **Swap** this value with the value at index 0. Now find the second smallest and **swap** that value into index 1. Continue until all values are in the correct order.
2. Create a new array. For each value in the old array, determine its rank and then copy it into the new array. Copy the new array into toSort.
3. Create random permutations of the values in the old array until you produce one that is in the correct order.
4. Determine the smallest value in the array and assign it index 0. Now search for the smallest value + 1. If it is present, copy it into index 1. If it is not, search for smallest + 2. Continue writing values into the array until you've written a value into the index (size-1).

Alternately you can view a large list of sorting algorithms at https://en.wikipedia.org/wiki/Sorting_algorithm but remember that simple is better than fast for this lab.

7- There are 100 positive, integer values in the file lab10_data.txt. Create these array in your main function:

int unsortedOne[50];
int unsortedTwo[50];

Now load fifty values from the data file into the first array and fifty into the second.

8- Copy the following code into your main function to see if your algorithm is faster than Bubble Sort:

clock_t bubble;
clock_t mine;

bubble = clock();
bubbleSort(unsortedOne, 50);
bubble = clock() - bubble;

mine = clock();
mySort(unsortedTwo, 50);
mine = clock() - mine;

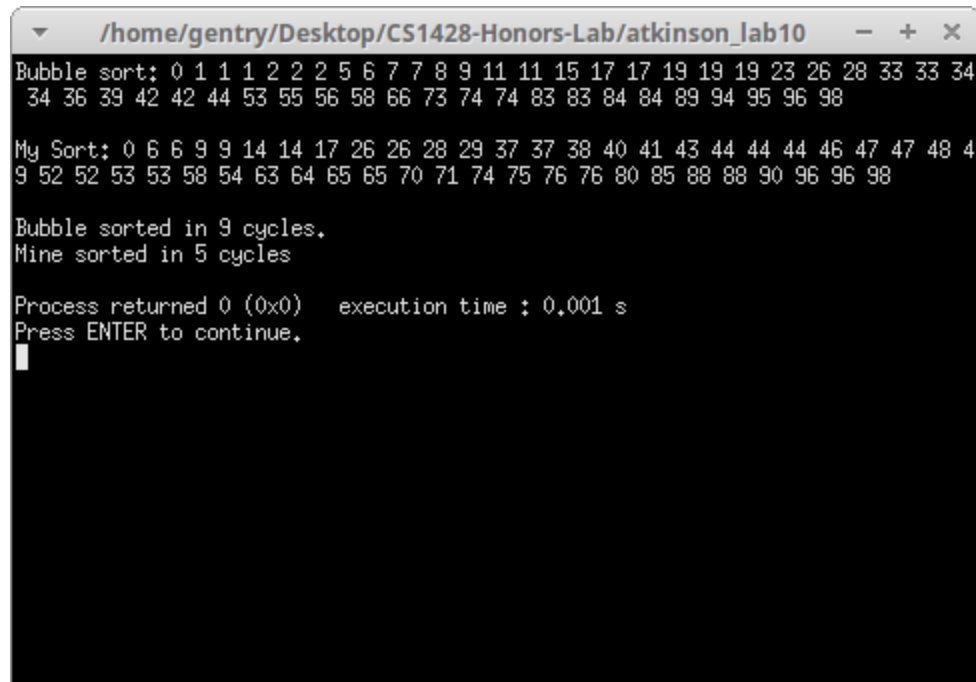**cout << "Bubble sort: ";**

```cpp
for(int i = 0; i < 50: i++)
        cout << unsortedOne[i] << " ";


cout << endl << "My Sort: ";
for (int i = 0; i<50; i++)
        cout<< unsortedTwo[i] << " ";


cout << endl << "Bubble sorted in " << bubble << " cycles." << endl;
cout << "Mine sorted in " << mine << " cycles" << endl;
```

9- Build and Run your code. Correct all errors. Your output should look something like this:

10- Submit your .cpp through TRACS. You can leave when you're done.