Assignment 1

Write three short programs that implement the Bubble Sort, Selection Sort, and Insertion Sort algorithms. Each of these programs will read up to 1000 integers, sort them, and then print them in a formatted fashion.

1. Initial Setup

1. Log in to Unix.
2. Run the `setup` script for Assignment 1 by typing:

```
setup 1
```

2. Input

Each program you write should read from standard input using input redirection. Input files for this assignment will contain a series of integers separated by whitespace.

3. Files We Give You

The setup script will create the directory `Assign1` under your `csci241` directory. It will copy a makefile named `makefile` to the assignment directory and will also create symbolic links to a number of sample data files that you can use to test your programs.

You should assume that we will test your programs using data files other than the ones supplied as part of this assignment.

Using the `makefile` we give you, you can build one specific program or build all three programs. To build a single specific program, run the command `make` followed by the target name for the program that you want to build (`bubble_sort`, `selection_sort`, or `insertion_sort`). For example:

```
z123456@turing:~$ make bubble_sort
```

To build all three programs, run the command `make all` or just `make`.

Running the command `make clean` will remove all of the object and executable files created by the `make` command.

The data files are named `random6.txt`, `random8.txt`, `random16.txt`, etc., with the numeric part of the file name specifying the number of random values that the file contains.

4. Files You Must Write

You will write three files for this assignment:

- `bubble_sort.cpp` - This file should contain your solution that uses the bubble sort algorthm.
- `selection_sort.cpp` - This file should contain your solution that uses the selection sort algorthm.

- `insertion_sort.cpp` - This file should contain your solution that uses the insertion sort algorthm.

The three files will obviously contain a great deal of identical code.

5. Output

The output should print 8 sorted numbers per line (of course the last line may have less than 8 numbers), nicely aligned in columns. Numbers should be right-aligned. Use the `setw` manipulator to print numbers with a width of 8 characters, padded with leading spaces. Print a newline character (`\n`) at the end of each line. The output should look like this:

```
z123456@turing:~/csci241/Assign1$ ./bubble_sort < random6.txt
      19       61       75       88       90       98
z123456@turing:~/csci241/Assign1$

z123456@turing:~/csci241/Assign1$ ./selection_sort < random16.txt
       7       15       15       20       30       37       42       52
      53       56       58       63       65       74       78       89
z123456@turing:~/csci241/Assign1$

z123456@turing:~/csci241/Assign1$ ./insertion_sort < random25.txt
       1       23      106      127      147      148      157      208
     239      265      282      483      561      576      579      677
     753      853      879      911      945      959      965      978
     982
z123456@turing:~/csci241/Assign1$
```

Note that your programs must *always* supply a newline character at the end of each line, even if the last line does not contain 8 numbers. In other words, after running your program, the Unix prompt must always appear in the leftmost column of the screen. Output that looks like this is **incorrect**:

```
z123456@turing:~/csci241/Assign1$ ./bubble_sort < random6.txt
      19       61       75       88       90       98z123456@turing:~/csci241/Assign1$
```

6. Hints

Pseudocode versions of the Selection Sort, Insertion Sort, and Bubble Sort algorithms are available on the course web site.

It may be easier to complete this assignment if you break it down into steps. Do one thing at a time, convince yourself that it works, and then move onto the next step. Below is an ordered sequence of steps that you might want to try for this assignment.

1. Decide which of the three sorting algorithms you're going to implement first. Selection sort is probably the easiest of the three to understand.

2. Write code to read the values and print them out. Don't bother storing the values in an array at this step and don't bother formatting the output - just read the numbers and print them out, one per line. Make sure that all of the numbers are being read and that you're not printing the final number in the file twice.

3. Add code to declare an array and store the values read in that array. Instead of printing each number as it is read, write code to print the values after they've all been stored in the array. Once

again, don't worry about formatting at this stage, just print one number one per line.

4. Implement the pseudocode for your chosen sorting algorithm in C++. Test and debug it using an input file with a small number of values. Check your output using the sample output above.

5. Format the output nicely.

6. Make sure to test your program with some of the other, larger data files.

7. For the other two sorting algorithms, make a copy of your completed source file and then modify the sort code in the copy. The input and output code will not need to be modified.

   If you're working on Unix, you can use the `cp` command to copy your source code file, e.g.:

```
z123456@turing:~/csci241/Assign1$ cp selection_sort.cpp bubble_sort.cpp
z123456@turing:~/csci241/Assign1$ cp selection_sort.cpp insertion_sort.cpp
z123456@turing:~/csci241/Assign1$
```