Purpose

The purpose of this assignment is for you to work with random numbers and sorting arrays.

Scenario

When an array is populated with data, more often than not the data is stored in a random manner depending on how the data is stored in the repository. There are several ways in which this data can be sorted in some manner that makes it efficient for future searches to be done with this data.

Problem

We are going to work with exactly 25 unsorted numbers to begin with. We first generate 25 random numbers in the range 1 to 100. As each number is generated we store them in an integer array (array size is 25). We print the contents of this initial array. You are likely to see that this array is unsorted since the numbers are randomly generated. We want to sort the array elements in ascending order where the first element has the smallest value and the last element has the largest. Sorting must be done using the Bubble Sort technique described below.

There are 3 source files that need to be copied from ~cs410c/public/11L to your local directory – main.c, functions.c and functions.h.

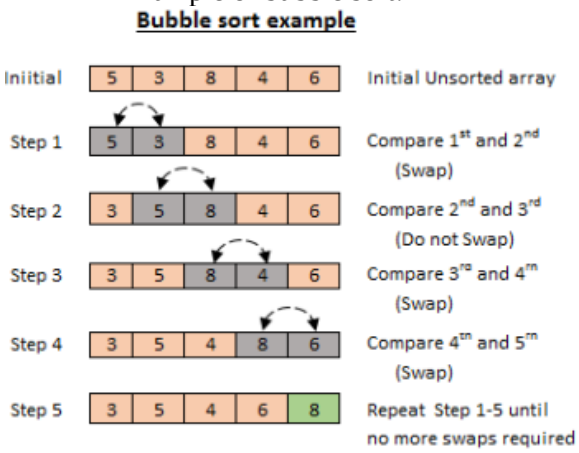
1. The main function has been written for you. **Do not change anything in main.c.** The function prototype definitions are made in functions.h. The stub for your function definitions are already made for you in functions.c. Your task is to fill in the functions with relevant code. Make sure you read the sequence of steps that are done in main.c.

2. The function fill_array() must generate as many random numbers as specified in the function argument 'size' and fill up the array being passed as 'array'. Make sure that the random numbers are in the range 1 to 100, look up the examples posted on the course web site under "Random Number Example" to see how you can get this to happen. You should use a random number "seed" which is different every time you run the program. If you need to include any other library files you must do so in functions.h which in turn is included in both the .c files.

3. The function print_array() must print the array contents one element at a time as shown in the output sample by using the arguments passed to it (array and its size).

4. Next we would like to sort the array using the following technique called bubble sort:

   Here's an example of bubble sort:



The function sort also takes two arguments: the array and its size and returns nothing.

5. Since we print the contents of the array before and after sorting you will hopefully see the sorted order of elements before the program ends. Your results will be different from mine since my randomly generated numbers are likely to be different from yours!

6. Submit all three files as 11L:

```
submit 11L main.c functions.c functions.h
```