Project 1

Bubble Sort

Due: Friday the 27th October 2017

Overview

In this project you will officially get your feet wet writing C code. This will give you practice with input and output, conditional execution, and looping constructs. Additionally, you will get some practice using Makefiles, the compiler, and potentially even the debugger.

Submission Instructions

This and all other assignments will be submitted through BBLearn. Look for the submission link in the same place you found this assignment.

Submit all your .c and .h[1] files along with your Makefile. Submit all of these files in a single submission, but do not zip them!

Note that your program may be evaluated by a program composed for just this purpose. As such, when the instructions indicate that something should be printed in a certain way, failing to do so may cause my evaluator to reject your program! Likewise, do not print anything that you are not instructed to print as this will also throw off the evaluator.

Technical Description and Instructions

Your program needs to be able to do only a few simple tasks:

1. Display a welcome message (terminated by three blank lines, see Welcome Message for details).
2. Take in some integer numbers from the user (up to 20) (see Reading In Numbers for details).
3. Sort the numbers given (see Bubble Sort for details).
4. Print the numbers in sorted order (see Program Output for details).

Additionally, be sure your Makefile produces an executable called "bubble_sort" so that the test script can run it automatedly.

Unless noted otherwise in the assignment, you can assume your user will behave. This means if you prompt them for a number, they won't enter "kitty-kat". This reduces the robustness of your program, but also reduces the amout of error checking you have to do.

Welcome Message

Your welcome message may be whatever text you want to display **except that it cannot contain three blank lines ('\n') in a row** as this is how the grading program will determine that your welcome message is finished. By dint of the same fact, your welcome **must end with three blank lines**.

[1] You may not have any .h files for some projects. That's ok!

Reading In Numbers

After displaying your welcome message, your program should then prompt the user to enter a number indicating how many numbers they will be providing to the program for it to sort. Do this by calling `printf()` with the the following format string:

```
Please enter how many numbers you want to sort (up to 20):\n\t
```

After printing the prompt, read in a single integer[2].

Your program should **verify that the number given is greater or equal to 2**. If it isn't, simply **printf the following format string and exit**[3]:

```
Since you have fewer than two numbers, they are already sorted!\n
```

Assuming that the above situation did not occur, your program should then **prompt the user to enter a number by calling `printf()` with this format string**:

```
Please enter the next number:\n\t
```

After printing the prompt, you should read in a single number from the user. Repeat this process a number of times equal to the quantity of numbers the user indicated they want to sort.

Do this a number of times equal to the number initially given by the user.

Bubble Sort

Your program will use bubble sort to sort the integers the user provides as the algorithm is simple and easy to implement. If you need a refresher on how bubble sort works, consider consulting the Wikipedia page for it.

Program Output

After taking in the last integer from the user, your program should simply **print out the following message**:

```
Here are the results:\n
```

Then, your program should **print out the numbers given in sorted order, with each number on a new line**. After this, your program should exit[4].

Things to Remember and Helpful Hints:

You should compile and run your program many times as you write it so that you can fix bugs as they arise rather than fixing them all at once at the end. The compiler is your guide showing you where you need to learn!

[2] You can use `scanf()` for this by passing it the format string "%i".
[3] That will require the use of the `exit()` function. Pass the constant `EXIT_SUCCESS` as the parameter
[4] You don't need to call `exit()` here, you can simply return from `main()`.

Grading Specification

For your submission to receive a grade of 'pass', it must fulfill the following criteria:

- **It must be submitted correctly**.
- I must be able to compile your program simply by invoking "make" in a directory containing your code and Makefile.
- The program must compile with no warnings[5] or errors.
- The program should run with no errors when given the correct inputs and should produce a correct result without crashing.

[5] This is with the `-Wall` and `-Wextra` warning flags passed to the compiler.