Project 2

Bubble Sort

Due: 10th October 2017

Overview

In this project you will explore a basic algorithm for sorting. This will give you additional practice with input and output and conditional execution as well as providing an opportunity to try for-loops for the first time. Additionally, you will get more practice using Makefiles and 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 files along with your Makefile, but do not zip them!

Note that your program may be evaluated by a program I compose 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!

Technical Description and Instructions

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

1. Display a welcome message (terminated by two 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).

For this project, you don't need to worry about the kind of edge case handling that we did for the previous project. You may assume that your user will only give you good inputs with the exception of giving a number less than two for how many numbers they intend to sort as noted in Reading In Numbers.

Additionally, be sure your makefile produces an executable called "bubble_sort" so that my test script can run it.

Welcome Message

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

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 with the following prompt:

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

Your program should **verify that the number given is greater than 2**. If it isn't, simply **print the following message and exit**:

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

Assuming that the user intends to enter more that two numbers, you should then **prompt the user to enter a number with this prompt**:

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

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. The pseudo-code given below shows how the algorithm works but **is not valid C code**. It assumes that the integers are stored in an array, called `data`, and `array_size` represents how many elements are in that array.

```
FOR completed_passes = 0, while it is less than array_size
  FOR current_index = 0, while it is less than (array_size - 1 - completed_passes)
    IF data[current_index] > data[current_index+1]
      temp = data[current_index]
      data[current_index] = data[current_index+1]
      data[current_index+1] = temp
    ENDIF
  ENDFOR
ENDFOR
```

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, and then exit**.

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 all at once at the end. The compiler is your guide showing you where you need to learn!

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 or errors.
- The program should run with no errors when given the correct inputs and should produce a correct result without crashing.