CSC 106 Assignment 2: Sorting through the bubbles, and in through the Or Gate

Marks: 60 marks.

Learning Goals:
At the end of this assignment you will be able to:

- Describe the Bubble Sort method of sorting a list of items;
- Determine which sort might work best in a given application;
- Design and create a circuit diagram using logic gate symbols;
- Create an animation in Scratch
- Create and update variables in Scratch
- Re-use code and update an interface in Scratch

Submission: This assignment may be done in groups of up to 4 people. You may also work individually if you prefer. The submitted assignment will have several components: The written component should be submitted in pdf format.
As well, you will have two Scratch file (.sb files) – one called Task4.sb, the other will be called myBubbler.sb

All files should be attached to the CSC106 Assignment 1 submission page on conneX.
A few submission notes:
---- Make sure that you include all group member's NAMES in the Assignment Text box on conneX. As well, please make sure that everyone's names are on the submitted file itself.
-- In the word processed document converted to a pdf, please make sure that you include the Task number/section you are answering above your response.
-- In the Scratch programs, make sure you have information in the project notes about what your programs do.
- MOST IMPORTANT: ONLY ONE ASSIGNMENT SUBMITTED PER GROUP!

Resources:

- Video overview of various sorts (Bubblesort is towards the end -- spoiler alert - they don't like it very well!): http://youtu.be/INHF_5RIxTE
- Overview of a number of sorts with animations (the descriptions are a bit technical, but the animations are good): http://www.sorting-algorithms.com/
- Comparison of sorts: http://www.cprogramming.com/tutorial/computersciencetheory/sortcomp.html
- Scratch reference: http://scratch.mit.edu/help/
- Scratch lessons (video): http://learnscratch.org/video-courses
- Scratch ideas: http://scratch.mit.edu/
- Installing Scratch v. 1.4 on your own computer: http://scratch.mit.edu/scratch_1.4/

TASK 1: Sorting (15 marks)

How does the sorting algorithm Bubble Sort work?

1. Research this algorithm and describe it *in your own words* (not copied from the research materials you found). Do not give pseudo code, but rather an overview of why and how the algorithm works. Feel free to use drawings or examples to support your answer. Provide references to any material you used to help come up with your description

2. Give pseudo code for Bubble Sort that matches your description in (1).

3. Consider this list: 5, 4, 3, 2, 1
How many comparisons in total would it take to sort this list into ascending order using the Bubble Sort you wrote pseudo code for? How many swaps would it take?

What if the list was 1, 2, 3, 4, 5
How many comparisons in total would it take to sort this list using Bubble Sort? How many swaps would it take?

What about 10, 3, 8, 2, 5
How many comparisons in total, how many swaps?

You can use the Bubbler to help you count the swaps, but you'll need to either have Scratch installed on your computer, or use Scratch on any of the lab computers.

TASK 1 DELIVERABLES: A word processed document containing the Task heading, the description (written in your own words), any diagrams you wish, and references/citations listed below the description. The pseudo code should be below the description.

Marks will be assigned as follows:

5 Marks for 1. (Remember to give your references in one of the styles discussed in lab. Be as clear as possible in your description. Don't copy and paste -- must be in your own words.)
5 Marks for 2. (Remember, state assumptions.)
5 Marks for 3. (I have not asked you to show your work, but don't just get the answer from someone else. This would be a great type of question for a midterm or final -- hint, hint -- so you should know how to approach a question like this. If you get help from someone, make sure you see how the comparisons and swaps work.)

TASK 2: Picking the right sort of sort. (5 marks)

You are part of a software design team creating a game for a mobile device (e.g. iPhone, tablet, iPad, etc.). It is a fantasy role playing game. The goal of the game is to develop characters of sufficient power so that the characters can take on ever more difficult quests. Any quest that a character undertakes will involve expending and replacing (hopefully) that character's power.

The game arena contains objects, and each object has an amount of power associated with it. When characters obtain these objects, the power associated with the object transfers to the character. Power allows characters to move about the fantasy realm, create structures and tools, overcome other characters, etc. -- essentially anything that helps the character solve the current quest.

The game space is full of power objects. Each object has an arbitrary power "value", so some objects are more powerful than others. Each character begins with 1 object. Each character can amass up to 10 objects. Every time a character gathers a new object a sorted list of that character's power objects is presented on the game screen, with the new object correctly inserted. It is a linear list, sorted in order of least powerful to most powerful.

Any time a character expends power and uses up the power of an object, or picks up a new power object the list is sorted and presented again. As well, a sorted list can be requested by its owner any time throughout the game.

Your task is to select an algorithm to handle the sorting. At this point you don't have to worry about coding, or defining data structures, but you do have to select a sorting algorithm to use at the back end. You narrow your selection down to two different comparison sorting algorithms that you think will do the job. You select two comparison sorts - Insertion Sort and Quick Sort

Create a table comparing the two algorithms. Make sure to compare how objects with the same power value are handled. As well, compare space requirements, worst case number of comparisons, and number of swaps (if applicable). As a final point, compare the ease of programming the two different sorts.

Finally, write no more than a paragraph noting which one of the two you recommend and why.

TASK 2 DELIVERABLES:
Using the same word processed document as you created for Task 1 make a new Task 2 heading.

3 marks for correctly comparing two algorithms.
2 marks for well reasoned final recommendations.

TASK 3: Logic Gate Design (20 marks)

Given the following formula:

((A AND B) OR (C AND NOT(A))) AND (A AND D)

1). Draw a diagram of this circuit using the logic gate symbols shown in the lab.
2). Fill in the truth table for this circuit

A	B	C	D	A⋀B	C⋀¬A	(A⋀B) ⋁ (C⋀¬A)	A∧D	((A⋀B) ⋁ (C⋀¬A)) ⋀ (A∧D)

Use the notes from the 3rd lab to help you create the circuit. If you like, you can draw (neatly!) the circuit on paper, take a photo of it, or scan it, and insert it.

TASK 3 DELIVERABLES: Responses to this question in your word processed document. PDF format is preferred, but just about any word processed file format will work. Use the notes from the 3rd lab to help you create the circuit. If you like, you can draw (neatly!) the circuit on paper, take a photo of it, or scan it, and insert it into your word processed document.

5 marks for correct response in the table.

15 marks for correct diagram.

TASK 4: Scratch – basic animation (5 marks)

1). Create a Scratch animation with at least two different sprites that move on the Scratch stage. The two sprites should communicate with one another via the "Broadcast and / or Broadcast and wait" and "When I receive" control structures. The animation should start when the green flag is clicked.

2). Create two variables and, using one or more of the repeat structures, display the changing / changed values of the variables on the stage

TASK 4 DELIVERABLES: Using Scratch (the stand-alone version 1.4) attach a .sb file called Task4.sb to the assignment. Make sure your Scratch file has info in the project notes describing what your program does.

3 marks for a two different sprites that move and communicate on the Scratch Stage.

2 marks for the variables – 1 mark for the variables, 1 mark for use of at least one repeat structure.

TASK 5: Improving the Bubbler (15 marks)

1). The Bubbler is a Scratch program, written using Version 1.4 (standalone version – it is attached to the bottom of the assignment). It prompts the user to put five integer values in a list. Then, it prompts the user to guess how many swaps will occur when the populated list is sorted using Bubble Sort. The user is told whether her or his guess is correct, but this is not really enough. If the user is incorrect the feedback from the sprite is “Almost, but not quite right”. The sprite should also say, using a talking bubble (e.g. like when she says “Hello”) what the actual number of swaps was. You may use another sprite for this task if you like.

2). Create a variable – call it **comps** – to show how many comparisons are performed when the Bubbler sorts the list of five items. Let **comps** show up on the left hand of the stage.

3). Improve the user interface used in the Bubbler. You can improve as you wish, but there are a few requirements that must be met:
A). Another sprite should be added.

B). Some type of animation should be added

C). It should be more of a game – your choice as to how this might be accomplished.

D). Anything else that you think might improve the program.

E). Update the Project Notes (erase what is there already) with what you did and why.

TASK 5 DELIVERABLES: Using Scratch (the stand-alone version 1.4) attach a .sb file called myBubbler.sb to the assignment. myBubbler.sb will be based on theBubbler.sb

2 marks for updating the sprite code to display actual number of swaps in a sentence bubble.

3 marks for a variable called comps which shows up on the left side of the stage, and shows how many comparisons are performed after the sort is completed.

10 marks for improving the game (innovation and “fun” will be well rewarded – but the project notes must explain what you have done and why you chose to do it).

Final note -- the staff at the **Computer Science Assistance Center** are there to help you. They can't "do" the assignment for you, but they can help out with application problems, printer problems, understanding questions, saving documents, uploading and moving files, etc. Call on them for help.