HOMEWORK 4: Bubble Sort, and Input Forms

Due date: **Sunday, 912:59:59, Weekend after Wk 5**

Status: WORKING VERSION

Part A (60 pts.) Create a JavaScript program that implements the bubble sort sorting technique.

The purpose of this program is to implement a more complex repeat or looping structure that makes use of both **for loop** and **while loop** syntax, use of **conditional** statement, and to get further experience with determining loop and conditional tests for governing execution. You should make use of your browser debugger console features and thoroughly track the step-by-step execution of your code.

I. The ***requirements definition*** *for Part A*:

a) Has 3 files: "MyLastNameBubble.js," "bubscript01.html," "bubscript01.css," and "stub-4-bubscript01.js." The last three files will be provided for you on Chalk.

b) Has your name, creation and edit dates, and purposes entered at the top of each script, with comments on your significant code steps.

c) Implement a version that extends the provided array data structure from 5 numbers to 10 randomly generated numbers. (You may do more than 10 if you want. In that case you must extend the display table accordingly.) You must design the number displays such that the initial *random* order of the data shows in the top row and then the final *sorted* order in bottom row for displaying the numbers. The provided design has two rows of data cells, the top one for the original unsorted list, and the bottom for the fully sorted list. Both should have their proper numbers shown. That is the concept to be carried through on the assignment. You may enhance them if you wish, but it is not required.

d) Extra Credit: Figure out a way to eliminate superfluous comparison calculations that occur for numbers that have already been completely sorted out.

For full credit your JS program must accomplish a, b, & c.

II. ***Algorithm Construction*** => *design the computational strategy for your program*

Begin by thinking through the example below and a few others on paper. Then try to map your understanding of the swapping process into the pseudo code version of the bubble sort algorithm below.

The bubble sort works by iterating down a list to be sorted from the first element to the last, comparing *each* pair of elements and switching their positions if necessary. This process is repeated as many times as necessary, until the list is sorted. Since the worst-case scenario is that the list is in reverse order, such that the first element in the sorted list is the last element in the starting list. This case requires the most exchanges because each number has to be "bubbled" up the list.

Here is a simple example:

In specific, given a list 2,3,1,5,4 the process goes as follows in the first pass:

2 & 3 are compared and found to be in proper smaller to larger order and left as is.
2,3,1,5,4

Then 3 & 1 are compared and reversed to be in proper order.
2,1,3,5,4

Then 3 & 5 are compared and left as is.
2,1,3,5,4

Then 5 & 4 are swapped
2,1,3,4,5

Thus at the end of the first pass, the list is partially sorted to: 2,1,3,5,4.
The next pass would lead to 1,2,3,4,5 and the sorting would be complete

On the first pass the 3 and 1 would be compared and switched, then the 5 and 4. On the next pass, the 2 and 1 would switch, and the list would be in order.

Pseudo code Handler: (Think this through on paper first!)

window.onload="initAll" (use provided setup function -- "stub-4-bubscript01.js" – to populate HTML table with random numbers. Call the function named: bubSrt(), pass the appropriate data.)

```
function bubSrt() {

    Determine the number of entries on table list of integers and set counter to = n.
    Declare needed variable(s).

    Set "swap" variable to true.

    Outer Loop while new swaps have occurred, i.e., "swap = true".

        reset swap variable for having "swapped two numbers" to false.

        Nested Loop starting from the (first number) on list to the (last number)

            if (number in position i) >  [number in position i+1] then

                swap number in position i with number in position i+1
                    [takes three lines of code to do number swap]

                set swap variable = true

        close loop
    close loop
close function
```

Note the nested looping structure in the above pseudo code.

III. ***Program Composition*** => *translate your strategy into program statements and objects*

Begin building your program in a small step-by-step way by testing that your code does what it is supposed to along the way. Use the Alert box for temporary output to check variable contents. Better, use the Variable function in browser debug mode as it can be very helpful here.

> Work out code for a short list of generated numbers to be sorted. Start with the actual swapping loop and use the numbers to get it working. You'll need to think through a "swapping strategy" to actually reorder the numbers. This will get you a single pass through the list. Then you have to figure out how to get multiple passes through the list.

IV. ***Program Refinement*** => *Thoroughly test and revise your code to improve function and remove bugs at each step of the way. Make sure all your permissions are correct.*

V. ***Iterate steps I to IV until you have a fully functioning program.***

You will need to learn how to use these new programming expressions.

The commands:

while loop	repeat until a true condition becomes false
for loop	repeat for a determinate number of times
if	execute statements if a condition tests true

The properties:

length	get and set "the number of" items in an array
innerHTML	content property for elements

The Booleans:

true; false	values for a test condition

Data structure:

array	container for multiple variables

The W3Schools.com page can give you syntax and semantics for all these expressions.

Be sure to backup your work frequently in two ways: a) "save" after every significant change, and b) "save as" into a *separate*, *differently named* "milestone" copy whenever you get code solid that would be a real pain to lose.

Part B (40 pts.) Forms creation and JS to generate selection list of state abbreviations

I. The ***requirements definition*** for Part B:

Create an HTML file with form inputs for “first name,” “last name,” “Address,” “City,” “State,” “Zip,” “Email,” a pair of radio buttons with the choice between “Credit Card” or “PayPal,” a collection of 3 checkboxes with “I am a member,” “Please send the newsletter,” and “I am interested in preshow events.”

For the “State,” input form create a selection list with the 50 state abbreviations. This list should be constructed at the event for window loading. Generate the list of states for a selection input form from a text array containing all the abbreviations.

II. ***Algorithm Construction*** => *for generating a loop of state abbreviations*

Do this by using a “for” loop with correct initial variable, test condition, and increment. This means you generate the selection list “on the fly” at time of loading so that the user sees the State input as a selection list. Put things on the webpage in the order listed above. You are free to use whatever CSS design you want. We will build on some of this in the next assignment.