Name : ___________________ Student Number : __________________

Assignment 2 – Structures, Functions, and Arrays

1. Sorting Algorithm:

Requirements:

Write an application that conducts the following:

- Provides a menu that is looped for user selection
- The menu will allow the user to select:
  - Define Random Number List
  - Sort Number List
  - Sorting Statistics
  - Exit
- The "Define Random Number List" will:
  - The Define Random Number List will be conducted in its own function, but the data will be made available back in the main function (your choice of method based on the design)
  - Fill a fixed size integer array with 20 data points
  - All data points will be randomly assigned integers between 0 and 100
  - Every time this option is selected it will over-write the data in the array (if any)
  - Once populated display the randomly populated data to the screen once control is returned back to the main application (to demonstrate data has been populated)
- The "Sort Number List" will:
  - Ask the user which sorting algorithm to use, with options consisting of:
    - Selection Sort
    - Bubble Sort
    - Counting Sort
  - Each sorting algorithm will be contained and conducted in its own function. We do not want to over-write the array populated in the "Define Random Number List" so ensure that you send by value only. Thus, the data should be able to be sorted infinite times using the same source data if desired.
  - Your application should verify prior to attempting to sort that the "Define Random Number List" has been executed and that there is data to sort! If this has not been done provide a reminder to the user to conduct this operation first.
  - Each sort algorithm should be coded from scratch to sort low to high, no pre-built tools
  - The sorted data results should be displayed to the screen along with the original for comparison. You can also display the interim work to the screen (optionally) to see the sorting occur.
  - Each sorting algorithm should be timed (as discussed in class) with the run time displayed to the screen with the final solutions
- The "Sorting Statistics" will:
  - Display the sort time for each sort algorithm that has been executed. If it has not been executed indicate that it is N/A
  - If the user re-populates the data array by operating "Define Random Number List" re-set all sort time parameters
- The "Exit" will:
  - Terminate the application
  - Indicate to the user how many data sets were generated, how many sorting algorithms were executed, and what was the fastest sort overall with its associated time.

Name : ___________________ Student Number : __________________

Grading (Total 50 points):

- 4 points for code style (i.e. does it follow good coding practises, naming conventions etc.)
- 4 points for clarity of code (comments, white space, indents, etc).
- 4 points for the menu, looping, and exit capabilities
- 5 points for the “Define Random Number List” requirements
- 30 points for the “Sort Number List” requirements
- 3 points for the “Sorting Statistics” requirements

Sample: //sample user input provided below in red (The __ are not required, here for clarity)

```
Welcome to the Sorting Algorithm Tool.

Please make your selection from the following:
    1.  Define Random Number List
    2.  Sort Number List
    3.  Sorting Statistics
    4.  Exit

Your Selection: 2
____

Sorry no data has been provided. Please define random data first then sort it.
__
Please make your selection from the following:
    1.  Define Random Number List
    2.  Sort Number List
    3.  Sorting Statistics
    4.  Exit

Your Selection: 1
__
Please enter a randomisation key: 43
The random data is:     40, 37, 97, 97, 61, 55, 10, 43, 47, 43, 29, 24, 65, 47, 59, 92, 79, 3, 82, 7
__
Please make your selection from the following:
    1.  Define Random Number List
    2.  Sort Number List
    3.  Sorting Statistics
    4.  Exit

Your Selection: 2
__

What sorting algorithm would you like to use?
    1.  Selection Sort
    2.  Bubble Sort
    3.  Counting Sort

Your Selection: 2
___
```

Name : ___________________   Student Number : __________________

```
Original Data:   40, 37, 97, 97, 61, 55, 10, 43, 47, 43, 29, 24, 65, 47, 59, 92, 79, 3, 82, 7
Sorted Data:     3, 7, 10, 24, 29, 37, 40, 43, 43, 47, 47, 55, 59, 61, 65, 79, 82, 92, 97, 97
Sorting Time:    345 mseconds
___


Please make your selection from the following:
    1.  Define Random Number List
    2.  Sort Number List
    3.  Sorting Statistics
    4.  Exit

Your Selection: 3
___

Sorting Statistics:
    1.  Selection Sort          N/A
    2.  Bubble Sort             345 mseconds
    3.  Counting Sort           N/A
____

Please make your selection from the following:
    1.  Define Random Number List
    2.  Sort Number List
    3.  Sorting Statistics
    4.  Exit

Your Selection: 1
____

Please enter a randomisation key: 12
The random data is:     22, 55, 98, 32, 16, 21, 33, 17, 84, 38, 13, 46, 80, 96, 56, 45, 55, 25, 49, 0
__

Please make your selection from the following:
    1.  Define Random Number List
    2.  Sort Number List
    3.  Sorting Statistics
    4.  Exit

Your Selection: 3
___

Sorting Statistics:
    1.  Selection Sort          N/A
    2.  Bubble Sort             N/A     //because new data, no sort has occurred
    3.  Counting Sort           N/A
____
```

```
Please make your selection from the following:
      1.  Define Random Number List
      2.  Sort Number List
      3.  Sorting Statistics
      4.  Exit

Your Selection: 4
___

Thank you for using the application.

Number of data sets generated:                2
Number of sorting algorithms executed:        1
Fastest sort was the: Bubble sort             345 ms          //only one in our case

The application will now terminate.
```

Save the source for number 1 as "Assn2-Sorting-YourName"

2. Battery:

NB. This application assumes simplifications from the physical world

Requirements:

Write an application that conducts the following:

- Create a structure called *battery* which will be used to represent a battery with data members consisting of:
  - voltage (in Volts),
  - how much energy the battery is capable of storing (in Joules)
  - how much energy it is currently storing (in Joules) , and
  - the battery type (i.e. "NiCad")
  - display the battery characteristics to the screen once created in your application using a display function

- Create a function called "power_device" that:
  - Inputs:
    - current of an electrical device (Amps)
    - time the device is to be powered by the battery (seconds)
    - the battery structure
  - Returns:
    - the battery structure (how you return is up to you)

  - The function first determines whether the battery's energy reserve is adequate to power the device for the prescribed time (see the equations below), if so:
    - the function updates the battery's energy reserve by subtracting the energy consumed and then returns the value true (1). The battery level is printed to the screen.
    - Otherwise it returns the value false (0) and leaves the energy reserve unchanged with a message to the user that there was not enough energy in the battery

- Create a function called "max_time" that:
  - Inputs:
    - The battery structure
    - The current of an electrical device (in Amps)
  - Returns:
    - The number of seconds the battery can operate the device before it is fully discharged (assume ideal battery). This is to be printed to the screen. How you return is up to you.
  - This function does not change any of the battery's component values but will determine how long an item can be operated.
  - Use the display function to print the current energy level to the screen once the program is returned to main (looking to verify that it is updated rather than local only)

- Create a function called "recharge" that:
  - Sets the current energy level to the max energy level (ie. full charge). Display to the screen the current energy level once charged with a description that the battery has been charged from main using the display function.

- Your application should initially conduct the following operations automatically with the appropriate outputs printed to the screen to denote they have been completed:
  - Create a battery model that is 12 volts and can hold a maximum energy storage of 5 x 10^6 J.
  - When created the battery should be fully charged
  - With the battery power a 4 amp light for 15 minutes
  - With the remaining capacity of the battery determine how long the battery could operate an 8 amp light
  - Recharge the battery
  - Recalculate how long the battery could operate an 8 amp light now that it is fully charged

- At this point provide a menu to the user that allows them to conduct any/all power_device, max_time, recharge, or re-define the battery in a looped manner. The appropriate output(s) should still be printed to the screen ideally with the associated inputs that were used during the computations. The re-define will always over-write the existing battery, you do not need to work with more than the single structure during this execution. Ensure you have an exit control.

Use the following equations in your design:

$P = I*V$	P = Power in watts (W)	V = Voltage in Volts (V)
$W = P*t$	I = Current in Amps (A)	W = Energy in Joules (J)
	t = Time in seconds (s)	(re-arrange the equations as required)

Grading (Total 31 points):

- 3 points for code style (i.e. does it follow good coding practises, naming conventions etc.)
- 3 points for clarity of code (comments, white space, indents, etc).
- 3 points for the menu, looping, and exit capabilities
- 2 points for the implementation of the structure in its entirety
- 15 points for the described functions
- 3 points for display capabilities from main that clearly denote application flow, prompts, etc.
- 2 points for the initial operations being displayed to the screen prior to manual entry

Save the source for number 2 as "Assn2-Battery-YourName"