Assignment #2 - C Programming Basics

CMPSC311 - Introduction to Systems Programming
Fall 2013 - Prof. McDaniel

Due date: September 20, 2013 (11:59pm)

In this assignment you will develop a program to manage several data types, data structures and arrays. Please read the following instructions very carefully and perform the assignment per the instructions.

1. Login to your virtual machine. Install the `wget` and `pgpgpg` using the `apt-get` utility. Refer to the course notes for details on how to perform installation of programs.

2. From your virtual machine, download the starter source code provided for this assignment. To do this, use the `wget` utility to download the file off the main course website:
   `cmpsc311-f13/docs/assign2-starter.tgz.gpg`

3. Create a directory for your assignments and copy the file into it. Change into that directory.

   ```
   % mkdir cmpsc311
   % cp assign2-starter.tgz.gpg cmpsc311
   % cd cmpsc311
   ```

4. Decrypt, untar, and unzip the repository using the gpg and tar tools. (You will need to use the password sent to you via email to decrypt the file).

   ```
   % gpg assign2-starter.tgz.gpg
   % tar xvfz assign2-starter.tgz
   ```

   Once unpacked, you will have the following starter files in the assign2 directory: `cmpsc311-f13-assign2.c` and `a2Support.h`. `cmpsc311-f13-assign2.c` contains the main function which reads in values from standard input. The `a2support.h` partially defines functions that you are to implement in the course of this assignment (see below).

5. You are to complete the `cmpsc311-f13-assign2` program. The `cmpsc311-f13-assign2` program receives 15 float values, one per line. The code for reading those values from standard input are provided in the assignment source code starter file.

6. Complete the code in the `cmpsc311-f13-assign2.c` file. Places where code needs to be added are indicated by `???`. See in file comments for hints. The program shall perform the following functions as guided by the `main()` function:

   (a) Read in 15 float values from the terminal and place them in an array. Note the code to perform this step is already provided.

   (b) Compute an 15 array of integers by calling the `myRound` function defined below on each value of the float array read in. Each index should contain the rounded value of the same index in the float array, e.g., element zero of the integer array should contain the rounded value of the zeroth element of the float array, etc.

   (c) Print out the values of the each array on their line using the `printFArray` and `printIArray` functions.

   (d) Increment the odd elements of the float array by 10.2 and the even elements of integer array by 112 (zeroth item is even). The incrementing should be accomplished by calling the `incrementFloat` and `incrementInt` functions, respectively.

(e) Print the largest value of each array using the `largestFloat` and `largestInt` functions, respectively.

(f) Print out the values of the each array on their line using the `printFArray` and `printIArray` functions.

(g) Sort the arrays using the `bsortFloat` and `bsortInt` functions respectively.

(h) Print out the sorted arrays again using the `printFArray` and `printIArray` functions.

(i) Print out the histogram of the array values.

7. You are to create a new file `a2support.c`. This will include the code for each of the functions defined in `a2support.h`. You are also to complete the function definitions in `a2support.h`. These functions are defined in table 1.

   Note: You can refer to the Wikipedia page `http://en.wikipedia.org/wiki/Bubble_sort` for information on the algorithm, but you cannot copy any code.

8. Add comments to all of your files stating what the code is doing. Fill out the comment function header for each function you are defining in the code. A sample header you can copy for this purpose is provided for the main function in the code.

9. In the `assign2` directory, create a Makefile for the source code existing in this directory. You should contain as many of the make and gcc utility features as possible, but must at least include detailed comments, built-in variables (e.g., $@), compile and link flag variables, suffix rules. The grade for this part of the assignment will hinge on the sophisticated use of `make`, clarity and detail of comments, etc.

To turn in:

1. Create a GPG encrypted tarball file containing the `assign2` directory, source code and build files as completed above. Email the program to `mcdaniel@cse.psu.edu` and the section TA (listed on course website) by the assignment deadline (11:59pm of the day of the assignment). The tarball should be named `LASTNAME-PSUEMAILID-assign2.tgz.gpg`, where LASTNAME is your last name in all capital letters and PSUEMAILID is your PSU email address without the "@psu.edu". For example, the professor was submitting a homework, he would call the file `MCDANIEL-pdm12-assign2.tgz.gpg`.

   Use the same password you used to decrypt the original file to encrypt this file.

2. Before sending the tarball, test it using the following commands (in a temporary directory – NOT the directory you used to develop the code):

```
% gpg LASTNAME-PSUEMAILID-assign2.tgz.gpg
% tar xvzf LASTNAME-PSUEMAILID-assign2.tgz
% cd assign2
% make
... (TEST THE PROGRAM)
```

Bonus Points: Create a log utility function for all output. This log utility should receive print data and print it out to STDERR with a timestamp on each line. For example, a snippet of the above output might read:

```
Mon Sep  9 07:12:33 PDT 2013 *********************************************************************
Mon Sep  9 07:12:33 PDT 2013 Received and computed values
Mon Sep  9 07:12:33 PDT 2013  9.50  45.64 313.11 113.89  81.56 250.00  11.90 469.98 313.11
Mon Sep  9 07:12:33 PDT 2013 4.68  34.33 8013.55 -10.15  11.50  88.00
Mon Sep  9 07:12:33 PDT 2013    10     46    313    114     82    250     12    470    313      5     34   8014    -10     12     88
Mon Sep  9 07:12:33 PDT 2013 The largest element of the float array is 8023.75
Mon Sep  9 07:12:33 PDT 2013 The largest element of the int array is   8014
Mon Sep  9 07:12:33 PDT 2013 *********************************************************************
Mon Sep  9 07:12:33 PDT 2013 Altered values
...
```

Function	**Parameters**	**Description**
`printFArray`	A reference to the array of floats and a integer length of the array	This function prints out an array of floats on a single line. The display width should be the same for each value (see sample output).
`printIArray`	A reference to the array of integers and a integer length of the array	This function prints out an array of integers on a single line. The display width be the same for each value (see sample output).
`myRound`	A single float value.	This should return the nearest integer (round up at 0.5).
`incrementFloat`	This should receive a pointer to a float value and a float value to increment by.	The function should increment the value **in the function** and reutn void.
`incrementInt`	This should receive a pointer to a integer value and a integer value to increment by.	The function should increment the value **in the function** and reutn void.
`largestFloat`	This function should receive a reference to an array and the array length.	The function should return the largest float value of the array passed.
`largestInt`	This should receive a reference to an array and the array length	The function should return the largest integer value of the array passed.
`bsortFloat`	This function should receive a reference to the float array and the array length.	The function should sort the values in the array using a bubble sort.
`bsortInt`	This function should receive a reference to the integer array and the array length.	The function should sort the values in the array using a bubble sort.
`printCharline`	This functions should receice a single character and a integer count.	The character should be printed out a number of times equal the integer, followed by a newline. If the integer is > 80, then only 80 characters should be printed.
`doHistogram`	The function should receive both and integer and a float array, as well as the length of both.	The function should compute and draw a histogram of values. The X-axis should contain 10 labeled columns with values $< 50, 50-(< 100), 100-(< 150)..., 450+$. The Y-axis should display the number of values in each of the bins. The top Y-value should be one greater than the largest count for any bin size. The format should mirror that provided in the sample output and be clean and organized.

Table 1: Functions to define and implement.

Note: Like all assignments in this class you are prohibited from copying any content from the Internet or discussing, sharing ideas, code, configuration, text or anything else or getting help from anyone in or outside of the class. Consulting online sources is acceptable, but under no circumstances should *anything* be copied. Failure to abide by this requirement will result dismissal from the class as described in our course syllabus.

Sample Output

```
*********************************************************************
Received and computed values
  9.50  45.64 313.11 113.89  81.56 250.00  11.90 469.98 313.11   4.68  34.33 8013.55 -10.15  11.50  88.00
    10     46    313    114     82    250     12    470    313      5     34   8014    -10     12     88
The largest element of the float array is 8023.75
The largest element of the int array is   8014
*********************************************************************
Altered values
  9.50  55.84 313.11 124.09  81.56 260.20  11.90 480.18 313.11  14.88  34.33 8023.75 -10.15  21.70  88.00
   122     46    425    114    194    250    124    470    425      5    146   8014    102     12    200
*********************************************************************
Sorted values
-10.15   9.50  11.90  14.88  21.70  34.33  55.84  81.56  88.00 124.09 260.20 313.11 313.11 480.18 8023.75
     5     12     46    102    114    122    124    146    194    200    250    425    425    470   8014
*********************************************************************
Historgram of values
10
 9      *
 8      *
 7      *
 6      *           *
 5      *           *
 4      *           *                                         *
 3      *     *     *                                         *
 2      *     *     *                 *     *           *     *
 1      *     *     *     *     *     *     *           *     *
      <50  050+  100+  150+  200+  250+  300+  350+  400+ >=450
```