CSE 2312 Programming Assignment 3 – Fall 2020

Programming Lab Policies:

- Labs that fail to compile, or do not terminate correctly, will receive a zero.
- Labs that fail to compile, or do not terminate correctly, may not be resubmitted for a grade. This includes instances where students did not upload the correct file for grading.
- **Students must make a credible attempt to pass all programming labs to receive a passing grade in the course.**

Write ARM assembly implementations for all of the functions below. The functions must be present in a single .s file. Your function/procedure names must be identical to that presented below, as your implementations will be tested with generic C code used by the TAs.

All of your functions must return a value such that the program will run to completion with no segmentation faults. If a function cannot be successfully implemented, it still must return a valid value: **no function may be omitted**. Attempting to omit a function will result in a compile error.

Submit your assignment via the submission link on Canvas. The name of this file should be **lab#_lastname_loginID.s** where # is the number of the lab assignment. Example: If your name is John Doe and your login ID is jxd1234, your submission file name must be "lab#_Doe_jxd1234.s".

All questions worth ten points.

1. void sortDecendingInPlace (int16_t x[], uint32_t count)

// input: array (x) containing count entries
// output: array (x), with the values sorted in descending order

2. float sumF32(float x[], uint32_t count)

// returns the sum of the elements in an array (x) containing count entries

3. double prodF64(double x[], uint32_t count)

// returns the product of the elements in an array (x) containing count entries

4. double dotpF64(double x[], double y[], uint32_t count)

// returns the dot product of two arrays (x and y) containing count entries

5. float maxF32(float x[], uint32_t count)

// returns the maximum value in the array (x) containing count entries

6. double absSumF64 (double x[], uint32_t count)

// input: array (x) containing count double entries
// output: the absolute value of the sum of the elements in array (x) containing count entries

7. double sqrtSumF64(double x[], uint32_t count)

// input: array (x) containing count double entries
// output: the square root of the sum of the elements in array (x) containing count entries

8. char getDirection (BUSINESS business[], uint32_t index)

// input: array of BUSINESS structs and index of BUSINESS
// output: street direction of BUSINESS (e.g., 'N', 'S', 'E', 'W')

9. uint32_t getAddNo (BUSINESS business[], uint32_t index)

// input: array of BUSINESS structs and index of BUSINESS
// output: address number of business

10. char * getCity(BUSINESS business[], uint32_t index)

// input: array of BUSINESS structs and index of BUSINESS
// output: city name of BUSINESS

BUSINESS will be provided in the sample driver. For a list of VFP instructions, go here:
http://www.keil.com/support/man/docs/armasm/armasm_dom1361289934045.htm