EE 3223 (Fall 2015)

Assignment #2, Due: September 15 by 11:59:59 pm

Submission Procedure: Place your source files in a folder called: <LastNameFirstName>. Name the files in the folder with the respective problem numbers: e.g.: 01.cpp, 02.cpp, etc. Zip this folder to get a file called <LastNameFirstName>.zip. Upload this file to **blackboard** by the due date.

Warm-up before attempting these problems. That is, look-up (or make-up) some simple problems and solve them. I googled for "simple programming problems". There are plenty of websites. Here is one that I found: http://adriann.github.io/programming_problems.html. Here is one for the more daring: https://projecteuler.net/problems

Problem 1 (2 points). Suggested time: 10 minutes.

Implement a function called add_array that takes two float arrays, their respective sizes, computes the sum of respective indices, and stores the result in the corresponding index of the second array, and returns true or false indicating success or failure. (Note that the function overwrites the content of the second array with the sum.) Using a main function, demonstrate that it works.

Problem 2 (2 points). Suggested time: 15 minutes.

Write a function called is_valid_email that takes an array of characters representing an email id, the array's size, and returns true if it is a valid email id, else false. An email id is valid if it contains a sequence of characters followed by '@', followed by another sequence of characters, and ends with ".com". Assume that the array only contains alphabets, and special characters '@' and '.'. Do not use inbuilt string library functionalities. Using a main function, demonstrate that it works.

Problem 3 (2 points). Suggested time: 10 minutes.

Implement a function *bool is_fibonacci(int)* which determines if the input number is a number in the Fibonacci sequence. The first two numbers in the Fibonacci sequence are 0 and 1 respectively. Subsequent numbers are calculated by adding the previous two numbers in the sequence. A partial sequence is: <0, 1, 1, 2, 3, 5, 8, 13, 21, ….> Demonstrate using a main function that it works for sample inputs.

Problem 3 (4 points). Suggested time: 30 minutes.

Implement a function *int* purge_duplicates(int* arr, int arr_sz, int* sz)* that takes an array of integers and the array's size as its input. The input array to this function may contain duplicate integer values. The goal of the function is to return a different array that contains no duplicate values. The size of this duplicate-free array is stored in the address *sz*, which is the third variable in the function. Using a main function, demonstrate that it works.