EECS 211 – Spring 2015
Programming Assignment 2
Due: Friday, April 17th, 2015

A prime number is a positive integer that is evenly divisible only by itself and 1. The integer 5 is a prime number because when five is divided by any integer other than 1 and 5 there is a non-zero remainder. The integer 77, on the other hand, is evenly divisible by 7 and 11. The first four prime numbers are 2, 3, 5, and 7. Any positive integer other than 1 can be written as a product of prime numbers. For example, 77 = 7*11, and 40 = 2*2*2*5. Prime numbers and prime factorizations play an important role in computer science in the area of encryption.

If p_0,p_1,p_2,… are the prime numbers in order, then an arbitrary integer x (greater than 1) can be tested for being prime as follows.

- If p_0 evenly divides x, then x is NOT prime.
- Otherwise, if p_1 evenly divides x, then x is NOT prime.
- Continue testing with successive primes until reaching n such that $p_n*p_n > x$.
- If none of the primes from p_0 to p_{n-1} evenly divide x, then x is prime.

This method suggests storing the primes (at least the first finite number of primes) in an array and using a loop to conduct this test. The loop counts through the list of primes in order until either the end of the list is reached or a divisor of x is found or $p_n*p_n > x$. Note, if we only know the first N primes and x is larger than the square of the largest prime we know, then this method cannot determine whether x is prime or not.

In this assignment you will start with the assignment2starter.cpp file posted with this assignment. Your job is to add the four functions mentioned in the posted code (and discussed below). You need not alter anything in the *main* function, other than writing additional tests for yourself. It is written to test your functions. Posted along with this assignment, you'll also find a file called output.txt. It provides the output of my code with the functions implemented properly.

- isPrime. The first argument is an array of integers holding prime numbers in order. The second argument is the number of elements in the array that actually hold prime numbers. The third argument is an integer that is to be tested for being prime. This function
  - returns 0 if the third argument is 0 or negative or divisible by one of the numbers in the first argument array.
  - returns 1 if it is able to determine that x is prime.
  - returns –1 if it is not able to determine whether x is prime or not.
- genPrimes. This function fills the rest of the array of primes; that is, it generates the 5th , 6th , … primes up to MAX_NUMBER_OF_PRIMES. The argument is the array in which to store these primes. The second argument is the number of elements in the array that actually hold prime numbers already. This function should use isPrime to test candidate integers for being prime.

- genPrimeFactors. This function should compute the prime factors of its last argument. The first argument is the list of the first nump primes. The second argument should be filled in with the number of times each prime number divides the fourth argument. For example, if the last argument was 45, the prime factors are 3, 3, 5. (45 = 3*3*5) The second array should be filled in so that the element with index 1 (i..e, corresponding to the prime number 3, which is stored in the primes array in subscript position 1) set to 2, the element with index 2 (i.e., corresponding to prime number 5) set to 1, and all other elements set to 0.
- displayPrimeFactors. The arguments are the list of prime numbers, the array of prime factors, the number of items in each array and the integer. This function displays the prime factors in a format like the following:

  ```
  The prime factors of 45 are 3 3 5
  ```

I've included commented stubs of the functions you will write at the beginning of the file returning dummy values, i.e., before the main function.

Rename your cpp file to your_netid.cpp and submit it on canvas before the posted deadline.