CSCI-14 Assignment #9, the Sieve of Eratosthenes – due 4/17/18

Eratosthenes was a Greek philosopher, mathematician, and scientist who, among many other things, made a remarkably accurate measurement of the size of the world in 240 BC, by using a well, a tower's shadow, a soldier who paced the distance between these landmarks counting his paces, and geometry. He developed an algorithm for finding prime numbers we know today as the "Sieve of Eratosthenes". Your job is to implement his algorithm, which works like this:

a) Assume all integers are prime (mark all array a[] elements 'prime')
b) turn off 0 and 1 (set a[0] and a[1] to 'not prime')
c) for(n = 2; n <= max / 2; n++)
    if n is still marked prime
        turn off all multiples of n (set to "not prime") // this is a **simple** loop
    // endif
  // endfor
d) Any number left over (still marked 'prime' after all the passes) is prime.

Use Eratosthenes' algorithm to calculate and print out all the prime numbers less than or equal to 10000.

Set up an array of 10001 (indices 0 to 10000, so MAX = 10000) ints or bools and initially set them all to 1 or true (except for elements 0 and 1). The index of the element in the array is the number you are testing for being prime. The element itself just holds whether or not its index is prime.

Whenever you reach a multiple of your n (see step 'c' above), turn off the entry in the array (set it to 0 or false). When you reach n >= ½ the size of the table, you know the indices of all elements in the table which still contain true are prime.

After running the sieve, print out the prime numbers as indicated by the array, 8 to a line separated by tabs, with no trailing tab on any line. Do not try to print the prime numbers while you are running the sieve: do not mix tasks together in your code. Send your output to a file within the program, using direct file output (an ofstream), and email that to me with your source listing.

You may not a print trailing tab on any line. This includes the last line of the output, if it is not 8 numbers long.

You MAY NOT use modulus (%) for ANY purpose in this assignment. Nor may you try to emulate modulus by any combination of integer dividing, multiplying, adding and/or subtracting to get to a remainder. You MUST implement the sieve algorithm as indicated above. This complete program is less than one page of code.

(35 points)