**Lince Rumainum**

**Advanced Java**

**Dr. Goulden**

**Week 3 – Chapter 7 (Single-Dimensional Arrays)**

**HOMEWORK**

**SIEVE OF ERATOSTHENES**

The Greek mathematician Eratosthenes (276 BCE – 195 BCE) devised a method of finding prime numbers, as follows:

* Create a list of consecutive integers from 2 up to the desired limit.
* Beginning with 2, mark off every second integer on the list.
* Move to the next *unmarked* number (which will be 3) and mark off every third integer.
* Move to the next *unmarked* number (which will be 5) and mark off every fifth integer.
* Move to the next *unmarked* number (which will be 7) and mark off every seventh integer.
* Move to the next *unmarked* number (which will be 11) and mark off every eleventh int…
* Continue until the end of the list has been reached.

When completed, every unmarked entry in the list (beginning with 2) represents a prime number.

Implement the Sieve of Eratosthenes with a program which performs the following tasks:

* Display a friendly greeting to the user.
* Prompt the user for a start value > 1. We will use this later.
* Prompt the user for a stop value. This will be the size of the array – 1.
* Use the stop value to create an array of type boolean and set every value to true.
* Execute the Sieve of Eratosthenes algorithm (using false to mark off entries)
* Display the total number of primes found in the interval [start, stop].

What is the largest array size you can run using Java? If your program can handle arrays of larger than 2 GB, make sure you input the user’s integer as a long, not an int. If not, explain why.

**The largest array size is approximately 2,147,000,000 because the input is an integer, which have approximately 2,147,483,648 bytes of memory. The program can only handle arrays less than 2GB because anything more than that will cause an out of memory error due to no memory location available in the heap. The user’s input cannot be a long because the array size has to be an integer value or it will cause a compile error.**

Use your program to answer the following questions:

What is the largest sieve that your program can (in principle) complete?

What is the significance of that value? How could you overcome that limitation?

How long does (or would) it take your program (in seconds) to compute a sieve…

… up to 100,000? **9,592 – 0 second**

… up to 1,000,000? **78,498 – 0 second**

… up to 10,000,000? **664,579 – 0 second**

… up to 100,000,000? **5,761,455 – 4 seconds**

… up to 1,000,000,000? **50,847,534 – 32 seconds**

… up to 1,000,000,000,000? **The computer could not calculate. NumberFormatException due to a parseInt error since the maximum number for integer data type is approximately 2,147,483,648.**

What is the asymptotic performance of the Sieve of Eratosthenes? **O(n(logn)(loglogn))**

How many prime numbers exist in each of the ranges given above? Compute as many as

you can, and explain why some can’t be done, at least not in time for class next week.

Submit your Java source code using the Moodle drop box. Also write a brief report giving these answers and include it in your submission.