This homework is due in class on Wednesday, October 10.

1. The *Sieve of Erastothenes* is an algorithm for finding every prime number that is less than or equal to some specified upper bound, M. To apply the sieve, place all the numbers between 2 and M, inclusive, into a set, A. Then, apply the following procedure:

```
for i from 2 to M
    if i is in the set A
        remove all multiples of i, starting with 2*i, from A
```

At the end of this procedure, A contains all the primes, and only the primes, in the range 2 through M. To apply the sieve of Erastothenes for a large value of M, you need an efficient implementation of sets. The standard Java class *BitSet*, which is defined in the package `java.util`, can efficiently represent sets of non-negative integers. For an integer n, The constructor `new BitSet(n)` creates an object that can represent any subset of the set $\{0, 1, 2, \ldots, n-1\}$ using just one bit for each potential member of the set. If *numbers* is such an object then the methods that are defined for *numbers* include:

numbers.set(i) — add i to the set (if it is not already there).

numbers.clear(i) — remove i from the set (if it is there).

numbers.isSet(i) — test whether i is in the set.

Write a Java program that uses a *BitSet* and the Sieve of Erastothenes to find the number of primes in the range 2 through M, where M is a constant in the program. Furthermore, once you have the set of primes in this range, use it to answer the following question: What is the largest "gap" between successive prime numbers in this range? Another way of looking at this is as the longest sequence of consecutive non-primes in the given range. Run your program for $M = 100,000,000$. Turn in a printout of your program and a statement of your result for the number of primes and for the size of the largest gap. You do not need to include comments in your program.

Note that there are additional methods in the *BitSet* class, in addition to those listed above, that you might find useful. Check out the API.

(Your program would also work for $M = 1,000,000,000$, if you want to try it for that value, but for that large a set, it will need more memory than Java usually gets when it runs. To request the extra memory that you need, run the program with the command `java -Xmx150m Sieve` (where *Sieve* is the name of the program).)

2. Consider the two 32-bit integers n and m shown below. Compute the 32-bit integers $n \,\&\, m$ and $n \mid m$. What subsets of $\{0, 1, 2, \ldots, 31\}$ do n, m, $n \,\&\, m$, and $n \mid m$ correspond to?

$$n = 0000\ 0100\ 1001\ 0000\ 0001\ 1101\ 1111\ 0101$$
$$m = 0110\ 0101\ 0011\ 1100\ 1111\ 1001\ 1000\ 0001$$

3. (Section 2.2, Exercise 4) Let A be a subset of some universal set U. Show that $\overline{\overline{A}} = A$ and that $A \cup \overline{A} = U$.

4. (Section 2.3, Exercise 6) Suppose that x and y are binary numbers. Under what circumstances will the binary numbers $x + y$ and $x \mid y$ be the same?