Section Handout 6

Based on a handout by Eric Roberts and Mehran Sahami

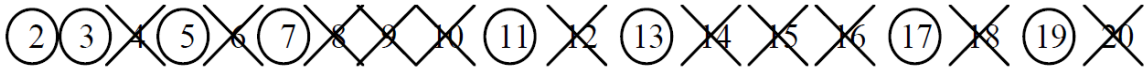
Problem One: The Sieve of Eratosthenes

In the third century B.C., the Greek astronomer Eratosthenes developed an algorithm for finding all the prime numbers up to some upper limit *N*. To apply the algorithm, you start by writing down a list of the integers between 2 and *N*. For example, if *N* were 20, you would begin by writing down the following list:

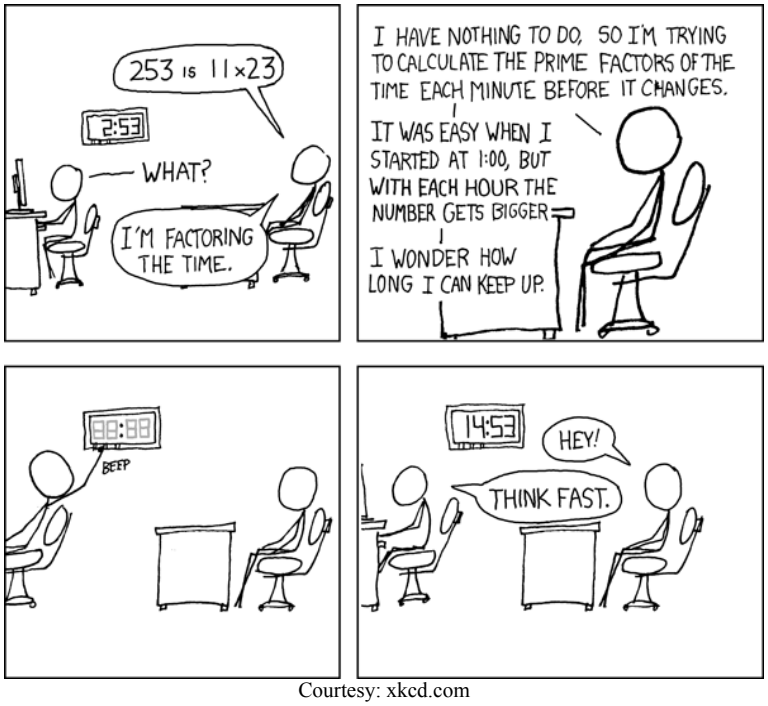
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

You then begin by circling the first number in the list, indicating that you have found a prime. You then go through the rest of the list and cross off every multiple of the value you have just circled, since none of those multiples can be prime. Thus, after executing the first step of the algorithm, you will have circled the number 2 and crossed off every multiple of two, as follows:

From here, you simply repeat the process by circling the first number in the list that is neither crossed off nor circled, and then crossing off its multiples. Eventually, every number in the list will either be circled or crossed out, as shown in this diagram:



The circled numbers are the primes; the crossed-out numbers are composites. This algorithm for generating a list of primes is called the sieve of Eratosthenes. Write a program that uses the sieve of Eratosthenes to generate a list of all prime numbers between 2 and 1000.



Courtesy: xkcd.com

Problem Two: Inverting Colors

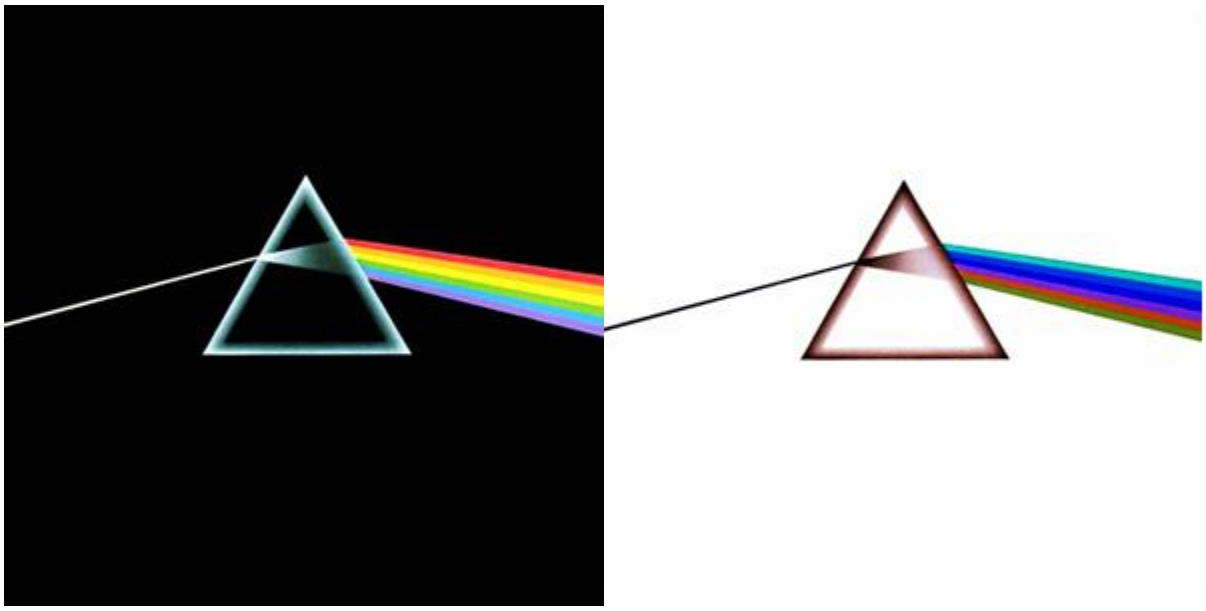
If you'll recall from our lecture on arrays, **`GImage`**s are internally represented on the computer as a two-dimensional array of pixels. Each pixel has a red, green, and blue component associated with them, and these values range from 0 (zero intensity) to 255 (maximum intensity). For example, pure red would have red value 255, green value 0, and blue value 0.

Given a **`GImage`**, we can construct the inverse of that **`GImage`** by flipping the intensity of each color channel. For example, if we have a pixel with red value 0, green value 255, and blue value 100, the inverse of that pixel would have red value 255, green value 0, and blue value 155. Visually, this represents taking the opposite of each of the colors in the image, so black pixels become white, green pixels become bright purple, etc.

Write a method

```
private GImage invertImage(GImage toInvert)
```

that accepts as input a **`GImage`** and produces a new **`GImage`** that's the inverse of the original **`GImage`**. As an example, here's a before-and-after comparison of the cover of Pink Floyd's "The Dark Side of the Moon" cover (would that make it the light side of the moon?)



Problem Three: Find the Median

The midterm has been graded, and it's your job to find the median score on the exam. The median score is defined as the score for which half of the scores on the exam are less than or equal to the median and half the exam scores are greater than or equal to the median. For example, if the exam scores are

```
30  42  38  45  42  32  37
```

then the median would be 38, because four of the numbers are no bigger than 38 and four of the numbers are no less than 38. You can assume that the number of scores on the exam is odd, so don't worry about the case where there are an even number of scores.

Write a method

```
private int medianOf(int[] scores)
```

that accepts as input an array of midterm scores (which range from 0 – 50, inclusive) and outputs their median. Again, you can assume the number of scores is odd.