

Section Handout #6: More Arrays and HashMaps

Portions of this handout by Eric Roberts

Please prepare the problem marked with **** prior to section.

1. How Prime!

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

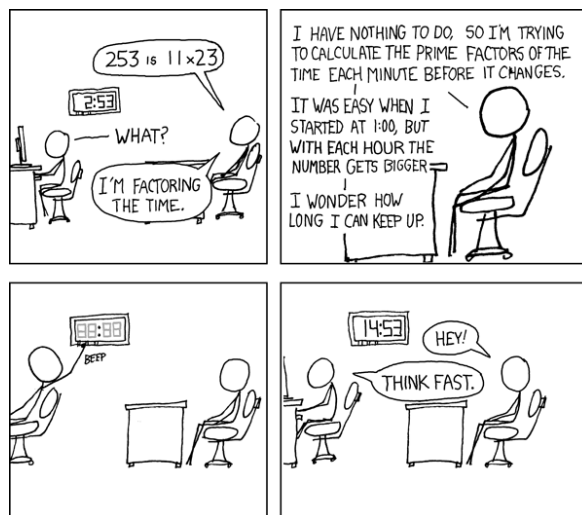
You then begin by circling the first number in the list, indicating that you have found a prime. You then go through the rest of the list and cross off every multiple of the value you have just circled, since none of those multiples can be prime. Thus, after executing the first step of the algorithm,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

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

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:

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

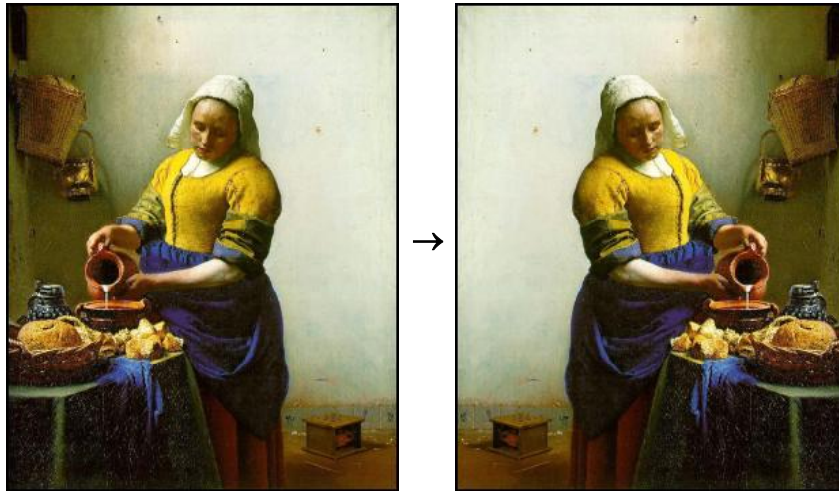
The circled numbers are the primes; the crossed-out numbers are composites. This algorithm for generating a list of primes is called the sieve of Eratosthenes. Write a program that uses the sieve of Eratosthenes to generate a list of all prime numbers between 2 and 1000.



Courtesy: xkcd.com

2. Image processing (Chapter 11, exercise 12, page 458)

Write a method `flipHorizontal` that reverses a picture in the horizontal dimension. Thus, if you had a `GImage` containing the image on the left (of Jan Vermeer's *The Milkmaid*, c. 1659), calling `flipHorizontal` on that image would return a new `GImage` as shown on the right:



**** 3. Name Counts ****

Write a program that asks the user for a list of names (one per line) until the user enters a blank line (i.e., just hits return when asked for a name). At that point the program should print out *how many times* each name in the list was entered. You may find that using a `HashMap` to keep track of the information entered by user may greatly simplify this problem. A sample run of this program is shown below.

```
CountNames
File Edit
Enter name: Alice
Enter name: Bob
Enter name: Alice
Enter name: Chelsea
Enter name: Don
Enter name: Don
Enter name: Don
Enter name: Chelsea
Enter name:
Entry [Chelsea] has count 2
Entry [Alice] has count 2
Entry [Bob] has count 1
Entry [Don] has count 3
```