Colin Kincaid
CS 106A

Section #5
July 23, 2018

Section Handout #5: Arrays

Portions of this handout by Eric Roberts and Marty Stepp

1D Arrays

1. Warmup: Trace

Write the final array contents when the following method is passed each array below:

```
private void mystery(int[] nums) {
    for (int i = 0; i < nums.length - 1; i++) {
        if (nums[i] > nums[i + 1]) {
            nums[i + 1]++;
        }
    }
}
```

```
Array 1: {10, 8, 9, 5, 5} Array 2: {12, 11, 10, 10, 8, 7}
```

2. Index Of

Write a method named **`indexOf`** that returns the index of a particular value in an array of integers. The method should return the index of the first occurrence of the target value in the array. If the value is not in the array, it should return **`-1`**. For example, if an array stores the following values:

```
int[] a = {42, 7, -9, 14, 8, 39, 42, 8, 19, 0};
```

Then the call **`indexOf(a, 8)`** should return **`4`** because the index of the first occurrence of value **`8`** in the array is at index **`4`**. The call **`indexOf(a, 2)`** should return **`-1`** because value **`2`** is not in the array.

3. Unique Numbers

Write a method named **`numUnique`** that accepts an array of integers as a parameter and returns the number of unique values in the array. The array is guaranteed to be in sorted order, which means that duplicates will be grouped together. For example, if an array stores the following values:

```
int[] list = {5, 7, 7, 7, 22, 22, 23, 35, 35, 40, 40, 40}
```

then the call **`numUnique(list)`** should return 6 because this list has 6 unique values **`(5, 7, 22, 23, 35, 40)`**. It is possible that the list might not have any duplicates. If **`list`** instead stored:

```
int[] list = {1, 2, 11, 17, 24, 25, 26, 31, 34, 37, 40, 41}
```

then a call on the method would return 12 because this list contains 12 different values. If passed an empty list, your method should return 0.

4. Banish

Write a method named **`banish`** that accepts two arrays of Strings a1 and a2 as parameters and removes all occurrences of a2's values from a1. An element is "removed" by shifting all subsequent elements one index to the left to cover it up, placing an empty string "" into the last index. The original relative ordering of a1's elements should be retained. For example, suppose the following two arrays are declared and the following call is made:

```
int[] a1 = {"a", "c", "a", "z", "a", "a", "p", "j", "w"};
int[] a2 = {"a", "p", "j"};
banish(a1, a2);
```

After the call has finished, the contents of a1 should become:

```
["c", "z", "w", "", "", "", "", "", ""]
```

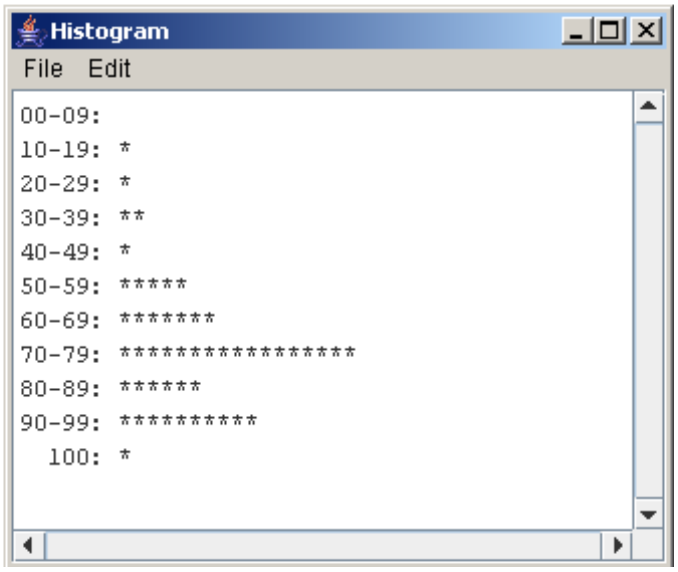
Notice that all occurrences of the values "a", "p", and "j" have been removed and replaced by empty strings at the end of the array, and the remaining values have shifted left to compensate.

5. Collapse

Write a method named **`collapse`** that accepts an array of integers as a parameter and returns a new array containing the result of replacing each pair of integers with the sum of that pair. For example, if an array called list stores the values **`{7, 2, 8, 9, 4, 13, 7, 1, 9, 10}`**, then the call of **`collapse(list)`** should return a new array containing **`{9, 17, 17, 8, 19}`**. The first pair from the original list is collapsed into 9 (7 + 2), the second pair is collapsed into 17 (8 + 9), and so on. If the list stores an odd number of elements, the final element is not collapsed. For example, if the list had been **`{1, 2, 3, 4, 5}`**, then the call would return **`{3, 7, 5}`**. Your method should not change the array that is passed as a parameter.

6. Histograms

Write a program that reads a list of exam scores from the file MidtermScores.txt (which contains one score per line) and then displays a histogram of those numbers, divided into the ranges 0-9, 10-19, 20-29, and so forth, up to the range containing only the value 100. If, for example, MidtermScores.txt contains the data shown in the right margin, your program should then be able to generate a histogram that looks as much as possible like the following sample run:



MidtermScores.txt

```
73
58
73
93
82
62
80
53
93
52
92
75
65
95
23
100
75
38
80
77
92
60
98
95
62
87
97
73
78
72
55
58
42
31
78
70
78
74
70
60
72
```

7. 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 10, you would begin by writing down the following list:

2 3 4 5 6 7 8 9 10

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.

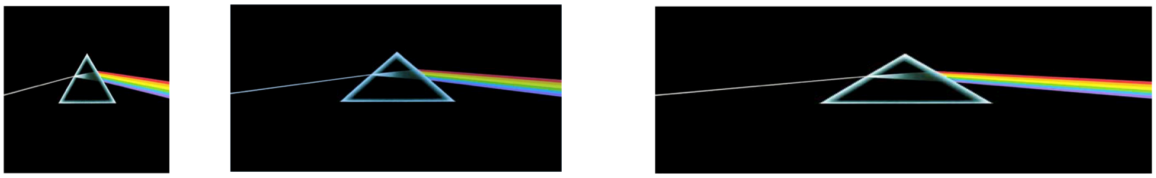
Images and Pixels (2D Arrays)

8. Flip Vertical

Write a method **`flipVertical`** that reverses a picture in the vertical dimension. Thus, if you had a **`GImage`** containing the image on the left (of Frida Kahlo's *Self-Portrait with Thorn Necklace and Hummingbird*, c. 1940), calling **`flipVertical`** on it would modify the pixels of that image as shown on the right:

9. Stretch

Write a method stretch that takes a `GImage` and an `int` representing a scale factor as parameters, and modifies the image to horizontally stretch it by the given factor. For example, if the factor is 2, stretch the image to be twice as wide, or if the factor is 3, stretch it to be 3x as wide. As an example, here's a before-and-after comparison of the cover of Pink Floyd's "The Dark Side of the Moon" stretched by 2x and 3x:



10. Trace

Consider the following method.

```
private void mystery2D(int[][] numbers) {
    for (int r = 0; r < numbers.length; r++) {
        for (int c = 0; c < numbers[0].length - 1; c++) {
            if (numbers[r][c + 1] > numbers[r][c]) {
                numbers[r][c] = numbers[r][c + 1];
            }
        }
    }
}
```

If a two-dimensional array numbers is initialized to:

```
3 4 5 6
4 5 6 7
5 6 7 8
```

...what are its contents after the call of `mystery(numbers);` ?