Alisha Adam & Rohit Talreja
CS 106A – Summer 2016

Solutions for Section #5

1. Index Of

```
1 private int indexOf(int[] list, int target) {
2     for (int i = 0; i < list.length; i++) {
3         if (list[i] == target) {
4             return i;
5         }
6     }
7     return -1;
8 }
```

2. Unique Numbers

```
1 private int numUnique(int[] list) {
2     if (list.length == 0) {
3         return 0;
4     }
5
6     int count = 1;
7     for (int i = 1; i < list.length; i++) {
8         if (list[i] != list[i - 1]) {
9             count++;
10        }
11    }
12    return count;
13 }
```

3. Collapse

```
1 public int[] collapse(int[] a) {
2     int[] result = new int[a.length / 2 + a.length % 2];
3     for (int i = 0; i < result.length - a.length % 2; i++) {
4         result[i] = a[2 * i] + a[2 * i + 1];
5     }
6     if (a.length % 2 == 1) {
7         result[result.length - 1] = a[a.length - 1];
8     }
9     return result;
10 }
```

4. Banish

```
1 public void banish(int[] a1, int[] a2) {
2     for (int i = 0; i < a1.length; i++) {
3         boolean found = false; // see whether a1[i] is contained in a2
4         for (int j = 0; j < a2.length && !found; j++) {
5             if (a1[i] == a2[j]) {
6                 found = true;
7             }
8         }
9         if (found) { // shift all elements of a1 left by 1
10            for (int j = i + 1; j < a1.length; j++) {
11                a1[j - 1] = a1[j];
12            }
13            a1[a1.length - 1] = 0;
14            i--; // so that we won't skip an index
15        }
16    }
17 }
```

5. Find Median

```
1 private static final int MAX_SCORE = 50; // max possible score
2
3 // Given a list of an odd number of midterm scores, returns the median score.
4 private int findMedian(int[] scores) {
5     double halfTheEntries = scores.length / 2.0;
6     int[] histogram = histogramFor(scores);
7     int cumulativeTotal = 0;
8     for (int i = 0; i <= MAX_SCORE; i++) {
9         cumulativeTotal += histogram[i];
10        if (cumulativeTotal >= halfTheEntries)
11            return i;
12        }
13    return 0; // can't get here, but Java requires us to return a value
14 }
15
16 // Given a list of scores, returns an array whose ith element is the number
17 // of exams that scored exactly i.
18 private int[] histogramFor(int[] scores) {
19    int[] result = new int[MAX_SCORE + 1];
20    for (int score: scores) {
21        result[score]++;
22    }
23    return result;
24 }
```

6. The Sieve of Eratosthenes

```
1 import acm.program.*;
2
3 /* Computes prime numbers using the Sieve of Eratosthenes */
4
5 public class SieveOfEratosthenes extends ConsoleProgram {
6 // The value up to which we should find prime numbers.
7     private static final int UPPER_LIMIT = 1000;
8
9     public void run() {
10    // Create an array of booleans that track whether or not we have crossed
11    // off each number. Initially, each number has not been crossed off, so
12    // we want the booleans to all be false. Since this is what Java does
13    // anyway, we don't need to explicitly set the boolean values to false.
14
15        boolean[] crossedOff = new boolean[UPPER_LIMIT + 1];
16
17        for (int n = 2; n <= UPPER_LIMIT; n++) {
18            if (!crossedOff[n]) {
19                println(n);
20
21                // Cross off all the multiples of n.
22                for (int k = n; k <= UPPER_LIMIT; k += n) {
23                    crossedOff[k] = true;
24                }
25            }
26         }
27    }
28 }
```