CS335 Exercise Set 1
Due Sunday 8/30

Instructions: Write Java solutions to the following problems.

Submit your solutions as zipped (or tar/gzip) IntelliJ project archives to the Canvas page for the course. Follow these *specific* naming and submission requirements for the problems in this exercise. First, name the IntelliJ project for problem (1) "abundant". Second, name the IntelliJ project for problem (2) "sieve". Third, zip those projects into one single file for submission: "exercise1.zip". Upload that file (exercise1.zip) to the canvas system for your submission to exercise 1.

The zip command to create your zip file for submission (from the terminal) is this:

```
zip -r exercise1.zip abundant sieve
```

[Motivation for specific naming conventions: this allows us to grade more efficiently and consistently because we can point the software we use at files that have consistent and predictable names across all students and assignments.]

1. An integer number is said to be *abundant* if the sum of its proper divisors, including 1 (but not the number itself) are greater than the number. For example, 24 is abundant, because the proper divisors of 24 are 1, 2, 3, 4, 6, 8, and 12. The sum of those proper divisors is 36, which is larger than 24. The "abundance" of 24 is (36-24) = 12.

   Write a method `abundant` and use it to identify all *odd* abundant numbers less than 10,000. Display the results.

2. (*Sieve of Eratosthenes*) A prime number is any integer greater than one that is evenly divisible only by itself and 1. The Sieve of Eratosthenes is a method of finding prime numbers. It operates as follows:
   a. Create a primitive type `boolean` array with all elements initialized to `true`. Array elements with prime indices will remain `true`. All other array elements will eventually be set to `false`.
   b. Starting with array index 2, determine whether a given element is `true`. If so, look through the remainder of the array and set to `false` every element whose index is a multiple of the index for the element with value `true.` Then continue the process with the next element with value `true`. For array index 2, all elements beyond element 2 in the array that have indices which are multiples of 2 (4, 6, 8, 10, etc.) will be set to `false`; for array index 3, all elements beyond element 3 in the array that have indices which are multiples of 3 (indices 6, 9, 12, 15, etc.) will be set to `false`; and so on.

   When this process completes, the array elements that are still `true` indicate that the index is a prime number. Write a Java program that uses an array of 10,000 elements to compute the "sieve" of prime numbers, and then display the ten prime numbers closest to (but less than) 10,000.