Objectives. Write simple Java programs that involve the following:

1. Control flow (conditional and loop) statements.
2. One- and two-dimensional arrays.
3. Standard input.
4. Recursion.

Problem 1. (*Counting Primes*) Implement the static method `isPrime()` in `PrimeCounter.java` that takes an integer argument x and returns `true` if it is prime and `false` otherwise. Also implement the static method `primes()` that takes an integer argument N and returns the number of primes less than or equal to N. Recall that a number x is prime if it is not divisible by any number $i \in [2, \sqrt{x}]$.

```
$ javac PrimeCounter.java
$ java PrimeCounter 100
25
$ java PrimeCounter 1000000
78498
```

Problem 2. (*Ramanujan's Taxi*) Srinivasa Ramanujan was an Indian mathematician who became famous for his intuition for numbers. When the English mathematician G. H. Hardy came to visit him one day, Hardy remarked that the number of his taxi was 1729, a rather dull number. To which Ramanujan replied, "No, Hardy! It is a very interesting number. It is the smallest number expressible as the sum of two cubes in two different ways." Verify this claim by writing a program `Ramanujan.java` that takes a command-line argument N and prints out all integers less than or equal to N that can be expressed as the sum of two cubes in two different ways. In other words, find distinct positive integers a, b, c, and d such that $a^3 + b^3 = c^3 + d^3$. Hint: Use four nested `for` loops, with these bounds on the loop variables: $0 < a \leq \sqrt[3]{N}$, $a < b \leq \sqrt[3]{N - a^3}$, $a < c \leq \sqrt[3]{N}$, and $c < d \leq \sqrt[3]{N - c^3}$; do not explicitly compute cube roots, and instead use `x * x * x < y` in place of `x < Math.cbrt(y)`.

```
$ javac Ramanujan.java
$ java Ramanujan 40000
1729 = 1^3 + 12^3 = 9^3 + 10^3
4104 = 2^3 + 16^3 = 9^3 + 15^3
13832 = 2^3 + 24^3 = 18^3 + 20^3
39312 = 2^3 + 34^3 = 15^3 + 33^3
32832 = 4^3 + 32^3 = 18^3 + 30^3
20683 = 10^3 + 27^3 = 19^3 + 24^3
```

Problem 3. (*Euclidean Distance*) Implement the static method `distance()` in `Distance.java` that takes position vectors x and y — each represented as a 1D array of doubles — as arguments and returns the Euclidean distance between them, calculated as the square root of the sums of the squares of the differences between the corresponding entries.

```
$ javac Distance.java
$ java Distance
5
-9 1 10 -1 1
5
-5 9 6 7 4
13.0
```

Problem 4. (*Matrix Transpose*) Implement the static method `tranpose()` in `Transpose.java` that takes a square matrix x — represented as a 2D array of doubles — as argument and transposes it in place.

```
$ javac Transpose.java
$ java Transpose
3 3
1 2 3
4 5 6
7 8 9
3 3
  1.00000   4.00000   7.00000
  2.00000   5.00000   8.00000
  3.00000   6.00000   9.00000
```

Problem 5. (*Exponentiation*) Implement the static method `power()` in `Power.java` that takes two integer arguments a and b and returns the value of a^b, computed recursively using the recurrence relation

$$a^b = \begin{cases} 1 & \text{if } b = 0, \\ aa^{b-1} & \text{if } b \text{ is odd}, \\ (a^2)^{b/2} & \text{if } b \text{ is even}. \end{cases}$$

```
$ javac Power.java
$ java Power 3 5
243
```

Files to Submit

1. `PrimeCounter.java`
2. `Ramanujan.java`
3. `Distance.java`
4. `Transpose.java`
5. `Power.java`

Before you submit:

- Make sure your programs meet the input and output specifications by running the following command on the terminal:

  ```
  $ python3 run_tests.py -v [<problems>]
  ```

  where the optional argument `<problems>` lists the problems (`Problem1`, `Problem2`, etc.) you want to test, separated by spaces; all the problems are tested if no argument is given.

- Make sure your programs meet the style requirements by running the following command on the terminal:

  ```
  $ check_style <program>
  ```

  where `<program>` is the `.java` file whose style you want to check.