Problem 1. (*Great Circle Distance*) Write a program called `GreatCircle.java` that accepts x_1 (double), y_1 (double), x_2 (double), and y_2 (double) as command-line arguments representing the latitude and longitude (in degrees) of two points on earth, and writes to standard output the great-circle distance (in km) between the two points, given by the formula

$$d = 6359.83 \arccos(\sin(x_1)\sin(x_2) + \cos(x_1)\cos(x_2)\cos(y_1 - y_2)).$$

```
>_ ~/workspace/exercise1
$ java GreatCircle 48.87 -2.33 37.8 -122.4
8701.387455462233
```

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

```
>_ ~/workspace/exercise1
$ java PrimeCounter 1000
168
```

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

```
>_ ~/workspace/exercise1
$ 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 accepts a matrix x — represented as a 2D array of doubles — and returns a new matrix that is the transpose of x.

```
>_ ~/workspace/exercise1
$ 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. (*Rational Number*) Implement an immutable data type called `Rational` that represents a rational number, ie, a number of the form a/b where a and $b \neq 0$ are integers. The data type must support the following API:

Rational	
`Rational(long x)`	constructs a rational number whose numerator is `x` and denominator is 1
`Rational(long x, long y)`	constructs a rational number given its numerator `x` and denominator `y` (†)
`Rational add(Rational other)`	returns the sum of this rational number and `other`
`Rational multiply(Rational other)`	returns the product of this rational number and `other`
`boolean equals(Object other)`	returns `true` if this rational number is equal to `other`, and false otherwise
`String toString()`	returns a string representation of this rational number

† Use the private method `gcd()` to ensure that the numerator and denominator never have any common factors. For example, the rational number 2/4 must be represented as 1/2.

```
>_ ~/workspace/exercise1
$ java Rational 10
a            = 1 + 1/2 + 1/4 + ... + 1/2^10 = 1023/512
b            = (2^10 - 1) / 2^(10 - 1) = 1023/512
a.equals(b) = true
```

Problem 6. (*Harmonic Number*) Write a program called `Harmonic.java` that accepts n (int) as command-line argument, computes the nth harmonic number H_n as a rational number, and writes the value to standard output.

$$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n-1} + \frac{1}{n}.$$

```
>_ ~/workspace/exercise1
$ java Harmonic 5
137/60
```

Files to Submit

1. `GreatCircle.java`
2. `PrimeCounter.java`
3. `Distance.java`
4. `Transpose.java`
5. `Rational.java`
6. `Harmonic.java`

Before you submit your files, make sure:

- You do not use concepts outside of what has been taught in class.
- Your code is adequately commented, follows good programming principles, and meets any specific requirements such as corner cases and running times.