Problem 1. (*Preparing for the Course*) Take care of the following action items (see course website for details):

- Sign up for Piazza using your UMass Boston email.
- Sign up for REEF software using your UMass Boston email.
- Sign up for Gradescope using your UMass Boston email.
- Setup the programming environment.
- Apply for a CS account.

Edit the Java program `CoursePrep.java` by replacing the ellipses (...) with relevant information and test the program by running the following command on the terminal:

```
$ java CoursePrep
Your UMass Boston email: jane.doe001@umb.edu
Your CS account username: jdoe
Have you read and understood the contents of the course website (yes/no)? yes
```

Problem 2. (*Wind Chill*) Given the temperature t (in Fahrenheit) and the wind speed v (in miles per hour), the National Weather Service defines the effective temperature (the wind chill) to be

$$w = 35.74 + 0.6215t + (0.4275t - 35.75)v^{0.16}.$$

Write a program `WindChill.java` that takes two doubles t and v as command-line arguments and writes the wind chill.

```
$ java WindChill 32 15
21.588988890532022
```

Problem 3. (*Day of the Week*) Write a program `DayOfWeek.java` that takes three integers m (for month), d (for day), and y (for year) as command-line arguments and writes the day of the week (0 for Sunday, 1 for Monday, and so on) $\mathcal{D}$, calculated as follows:

$$\begin{aligned} y_0 &= y - (14 - m)/12 \\ x_0 &= y_0 + y_0/4 - y_0/100 + y_0/400 \\ m_0 &= m + 12 \times ((14 - m)/12) - 2 \\ \mathcal{D} &= (d + x_0 + 31 \times m_0/12) \bmod 7 \end{aligned}$$

```
$ java DayOfWeek 3 14 1879
5
```

Problem 4. (*Great Circle*) Write a program `GreatCircle.java` that takes four doubles x_1, y_1, x_2, and y_2 representing the latitude and longitude in degrees of two points on earth as command-line arguments and writes the great-circle distance (in km) between them, given by the equation:

$$d = 111 \arccos(\sin(x_1)\sin(x_2) + \cos(x_1)\cos(x_2)\cos(y_1 - y_2)).$$

Note that this equation uses degrees, whereas Java's trigonometric functions use radians. Use `Math.toRadians()` and `Math.toDegrees()` to convert between the two. Use your program to compute the great-circle distance between Paris (48.87° N and 2.33° W) and San Francisco (37.8° N and 122.4° W).

```
$ java GreatCircle 48.87 -2.33 37.8 -122.4
8701.389543238289
```

Problem 5. (*Three Sort*) Write a program `ThreeSort.java` that takes three integers as command-line arguments and writes them in ascending order, separated by spaces. Use `Math.min()` and `Math.max()`.

```
$ java ThreeSort 1 2 3
1 2 3
$ java ThreeSort 1 3 2
1 2 3
$ java ThreeSort 2 1 3
1 2 3
$ java ThreeSort 2 3 1
1 2 3
$ java ThreeSort 3 1 2
1 2 3
$ java ThreeSort 3 2 1
1 2 3
```

Problem 6. (*Three Dice*) Write a program `ThreeDice.java` that writes the sum of three random integers between 1 and 6, such as you might get when rolling three dice.

```
$ java ThreeDice
5
```

Problem 7. (*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 $i \in [2, \sqrt{x}]$.

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

Problem 8. (*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 $c < d \leq \sqrt[3]{N - c^3}$.

```
$ 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 9. (*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.

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

Problem 10. (*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.

```
$ 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 11. (*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

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

```
$ java Power 3 5
243
```

Files to Submit

1. `CoursePrep.java`
2. `WindChill.java`
3. `DayOfWeek.java`
4. `GreatCircle.java`
5. `ThreeSort.java`
6. `ThreeDice.java`
7. `PrimeCounter.java`
8. `Ramanujan.java`
9. `Distance.java`
10. `Transpose.java`
11. `Power.java`

Before you submit:

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

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

  where the optional argument `<problems>` lists the problems (`Problem1`, `Problem2`, etc.) you want to test; 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.