Problem 1. (*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 `wind_chill.py` that takes two floats t and v as command-line arguments and writes the wind chill.

```
$ python wind_chill.py 32 15
21.5889888905
```

Problem 2. (*Body Mass Index*) The body mass index (BMI) is the ratio of the weight of a person (in kilograms) to the square of the height (in meters). Write a program `bmi.py` that takes two floats w (for weight) and h (for height) as command-line arguments and writes the BMI.

```
$ python bmi.py 75 1.83
22.3954134193
```

Problem 3. (*Polar Coordinates*) Write a program `polar.py` that takes two floats x and y representing the Cartesian coordinates of a point as command-line arguments and writes the corresponding polar coordinates $r = \sqrt{x^2 + y^2}$ and $\theta = \arctan(y/x)$.

```
$ python polar.py 1 1
1.41421356237
0.785398163397
```

Problem 4. (*Order Check*) Write a program `order_check.py` that takes three floats x, y, and z as command-line arguments and writes `True` if the values are strictly ascending or descending (ie, $x < y < z$ or $x > y > z$), and `False` otherwise.

```
$ python order_check.py 2 4 5
True
$ python order_check.py 2 7 6
False
$ python order_check.py 7 3 1
True
$ python order_check.py 7 3 4
False
```

Problem 5. (*Day of the Week*) Write a program `day_of_week.py` 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:

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

```
$ python day_of_week.py 3 14 1879
5
```

Problem 6. (*Mercator Projection*) The Mercator projection is a conformal (angle preserving) projection that maps latitude φ and longitude λ to rectangular coordinates (x, y). It is widely used — for example, in nautical charts and in the maps that you print from the web. The projection is defined by the equations $x = \lambda - \lambda_0$ and $y = \ln((1 + \sin\varphi)/(1 - \sin\varphi))/2$, where λ_0 is the longitude of the point in the center of the map. Write a program `mercator.py` that takes three floats λ_0, φ, and λ as command-line arguments and writes its projection, ie, the x and y values, separated by a space. Note that the equations use degrees, whereas Python's trigonometric functions use radians. Use `math.radians()` to convert degrees to radians. Use your program to compute the Mercator projection of Boston (42.36° N and 71.06° W) with the center of the map being the prime meridian (0°).

```
$ python mercator.py 0 42.36 -71.06
-71.06 0.817646151942
```

Problem 7. (*Great Circle*) Write a program `great_circle.py` that takes four floats 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 Python's trigonometric functions use radians. Use `math.radians()` and `math.degrees()` 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).

```
$ python great_circle.py 48.87 -2.33 37.8 -122.4
8701.38954324
```

Problem 8. (*Three Sort*) Write a program `three_sort.py` that takes three integers as command-line arguments and writes them in ascending order, separated by spaces. Use `min()` and `max()`.

```
$ python three_sort.py 1 2 3
1 2 3
$ python three_sort.py 1 3 2
1 2 3
$ python three_sort.py 2 1 3
1 2 3
$ python three_sort.py 2 3 1
1 2 3
$ python three_sort.py 3 1 2
1 2 3
$ python three_sort.py 3 2 1
1 2 3
```

Problem 9. (*Random Integer*) Write a program `random_int.py` that takes two integers a and b from the command line and writes a random integer between a (inclusive) and b (exclusive).

```
$ python random_int.py 10 20
13
```

Problem 10. (*Three Dice*) Write a program `three_dice.py` that writes the sum of three random integers between 1 and 6, such as you might get when rolling three dice.

```
$ python three_dice.py
5
```

Files to Submit

1. `wind_chill.py`
2. `bmi.py`
3. `polar.py`
4. `order_check.py`
5. `day_of_week.py`
6. `mercator.py`
7. `great_circle.py`
8. `three_sort.py`

9. `random_int.py`
10. `three_dice.py`
11. `report.txt`

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 [<problems>]
  ```

  where the optional argument `<problems>` lists the numbers of the problems 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:

  ```
  $ pep8 <program>
  ```

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

- Make sure your report doesn't exceed 400 lines, doesn't contain spelling mistakes, and doesn't contain lines that exceed 80 characters.