This document only contains the project problems. For the programming exercises on concepts needed for the project, please refer to the project checklist .

Problem 1. (*Day of the Week*) Write a program `day_of_week.py` that takes three integers m (month), d (day), and y (year) as command-line arguments and writes the day of the week (0 for Sunday, 1 for Monday, and so on) D, calculated as

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

```
$ python3 day_of_week.py 3 14 1879
5
```

Problem 2. (*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, with the center of map set to the prime meridian (0 degrees) and radius of Earth set to 1, is defined by the equations

$$x = \lambda \text{ and } y = \frac{\ln\left(\frac{1+\sin\varphi}{1-\sin\varphi}\right)}{2}.$$

Write a program `mercator.py` that takes two floats φ (latitude in degrees) and λ (longitude in degrees) as command-line arguments and writes the corresponding x and y values, separated by a space.

```
$ python3 mercator.py 42.36 -71.06
-71.06 0.8176461519422712
```

Problem 3. (*Great Circle Distance*) Write a program `great_circle.py` that takes four floats x_1, y_1, x_2, and y_2 (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

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

```
$ python3 great_circle.py 48.87 -2.33 37.8 -122.4
8701.389543238289
```

Problem 4. (*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.

```
$ python3 wind_chill.py 32 15
21.588988890532022
```

Problem 5. (*Gravitational Force*) Write a program `gravitational_force.py` that takes floats m_1 and m_2 representing the masses (in kg) of two objects and a float r representing the distance (in m) between their centers as command-line arguments and writes the gravitational force F (in N) acting between the objects, calculated as

$$F = G\frac{m_1 m_2}{r^2},$$

where $G = 6.674 \times 10^{-11}$ (in $\text{m}^3\ \text{kg}^{-1}\ \text{s}^{-2}$) is the gravitational constant.

```
$ python3 gravitational_force.py 2e30 6e24 1.5e11
3.5594666666666664e+22
```

Problem 6. (*Snell's Law*) Snell's law states that given two mediums, the ratio of the sines of the angles of incidence and refraction is equivalent to the reciprocal of the ratio of the indices of refraction of the two mediums, ie,

$$\frac{\sin(\theta_1)}{\sin(\theta_2)} = \frac{n_2}{n_1}.$$

Write a program `snell.py` that takes three floats θ_1 (in degrees), n_1, and n_2 as command-line arguments and writes the corresponding angle of refraction θ_2 in degrees.

```
$ python3 snell.py 58 1 1.52
33.912513998258994
```

Problem 7. (*Gambler's Ruin*) Consider a coin-flipping game with two players where player one wins each toss with probability p, and player two wins with probability $q = 1 - p$. Suppose player one has n_1 pennies and player two n_2 pennies. Assuming an unfair coin (ie, $p \neq 1/2$), the probabilities P_1 and P_2 that players one and two, respectively, will end penniless are

$$P_1 = \frac{1 - (\frac{p}{q})^{n_2}}{1 - (\frac{p}{q})^{n_1+n_2}} \text{ and } P_2 = \frac{1 - (\frac{q}{p})^{n_1}}{1 - (\frac{q}{p})^{n_1+n_2}}.$$

Write a program `gambler.py` that takes two integers n_1 and n_2 and a float p as command-line arguments and writes the probabilities P_1 and P_2, separated by a space.

```
$ python3 gambler.py 10 100 0.51
0.6661883734200654 0.3338116265799349
```

Problem 8. (*Waiting Time*) If λ is the average number of events per unit of time, the probability $P(t)$ that one has to wait longer than time t until the next event is given by the exponential distribution

$$P(t) = e^{-\lambda t}.$$

Write a program `waiting_time.py` that takes two floats λ and t as command-line arguments and writes the probability $P(t)$.

```
$ python3 waiting_time.py 0.1 5
0.6065306597126334
```

Problem 9. (*Die Roll*) Write a program `die_roll.py` that takes an integer n representing the number of sides of a fair die, rolls an (n-sided) die twice, and writes the sum of the numbers rolled.

```
$ python3 die_roll.py 6
12
```

Problem 10. (*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. Your solution must only use functions `min()` and `max()`.

```
$ python3 three_sort.py 3 1 2
1 2 3
```