Instructions

- You are not allowed to use concepts not covered in class or from sections beyond *Basic Data Types*.
- Make sure your code is clean, well-organized, uses meaningful variable names, includes useful comments, and is efficient.
- Make sure you edit the sections (`#1` mandatory, `#2` if applicable, and `#3` optional) in the given `notes.txt` file as appropriate. In section `#1`, for each problem, state its goal in your own words and describe your approach to solve the problem along with any issues you encountered and if/how you managed to solve those issues.

Goal: Implement simple programs *without* using control-flow (ie, branch and loop) statements.

Problem 1. (*Name and Age*) Write a program called `name_age.py` that receives `name` (str) and `age` (str) as command-line inputs, and writes the string "`name` is `age` years old." as standard output.

```
× ~/workspace/straightline_programs
$ python3 name_age.py Alice 19
Alice is 19 years old.
$ python3 name_age.py Bob 23
Bob is 23 years old.
```

Problem 2. (*Greet Three*) Write a program called `greet_three.py` that receives `name1` (str), `name2` (str), and `name3` (str) as command-line inputs, and writes the string "Hi `name3`, `name2`, and `name1`." as standard output.

```
× ~/workspace/straightline_programs
$ python3 greet_three.py Alice Bob Carol
Hi Carol, Bob, and Alice.
$ python3 greet_three.py Dan Eve Fred
Hi Fred, Eve, and Dan.
```

Problem 3. (*Day of the Week*) Write a program called `day_of_week.py` that receives `m` (int), `d` (int), and `y` (int) as command-line inputs representing a date, and writes the day of the week (0 for Sunday, 1 for Monday, and so on) `dow` as standard output. The value for `dow` is computed as

$$\begin{aligned}
\texttt{y0} &= \texttt{y} - (14 - \texttt{m})/12 \\
\texttt{x0} &= \texttt{y0} + \texttt{y0}/4 - \texttt{y0}/100 + \texttt{y0}/400 \\
\texttt{m0} &= \texttt{m} + 12 \times ((14 - \texttt{m})/12) - 2 \\
\texttt{dow} &= (\texttt{d} + \texttt{x0} + 31 \times \texttt{m0}/12) \bmod 7
\end{aligned}$$

```
× ~/workspace/straightline_programs
$ python3 day_of_week.py 3 14 1879
5
$ python3 day_of_week.py 2 12 1809
0
```

Problem 4. (*Three Sort*) Write a program called `three_sort.py` that receives `x` (int), `y` (int), and `z` (int) as command-line inputs, and writes the numbers (separated by a space) in ascending order as standard output. Your solution must only use `min()`, `max()`, and basic arithmetic operations to compute the ordering.

```
× ~/workspace/straightline_programs
$ python3 three_sort.py 1 3 2
1 2 3
$ python3 three_sort.py 3 2 1
1 2 3
```

Problem 5. (*Body Mass Index*) The body mass index (BMI) is the ratio of the weight `w` of a person (in kg) to the square of the height `h` (in m). Write a program called `bmi.py` that receives `w` (float) and `h` (float) as command-line inputs, and writes the BMI value as standard output.

```
× ~/workspace/straightline_programs
$ python3 bmi.py 75 1.83
22.395413419331717
$ python3 bmi.py 97 1.75
31.6734693877551
```

Problem 6. (*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

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

Write a program called `wind_chill.py` that receives `t` (float) and `v` (float) as command-line inputs, and writes the wind chill `w` as standard output.

```
× ~/workspace/straightline_programs
$ python3 wind_chill.py 32 15
21.588988890532022
$ python3 wind_chill.py 10 10
-3.5402167842280647
```

Problem 7. (*Gravitational Force*) Write a program called `gravitational_force.py` that receives `m1` (float), `m2` (float), and `r` (float) as command-line inputs, representing the masses (in kg) of two objects and the distance (in m) between their centers, and writes as standard output the gravitational force `f` (in N) acting between the objects, computed as

$$\mathtt{f} = \mathtt{G}\frac{\mathtt{m1m2}}{\mathtt{r}^2},$$

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

```
× ~/workspace/straightline_programs
$ python3 gravitational_force.py 2e30 6e24 1.5e11
3.5594666666666664e+22
$ python3 gravitational_force.py 6e24 7.35e22 3.84e8
1.9960083007812498e+20
```

Problem 8. (*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 $\mathtt{q} = 1 - \mathtt{p}$. Suppose player one has `n1` pennies and player two `n2` pennies. Assuming an unfair coin (ie, $\mathtt{p} \neq 1/2$), the probabilities `p1` and `p2` that players one and two, respectively, will end penniless are

$$\mathtt{p1} = \frac{1 - (\frac{\mathtt{p}}{\mathtt{q}})^{\mathtt{n2}}}{1 - (\frac{\mathtt{p}}{\mathtt{q}})^{\mathtt{n1+n2}}} \text{ and } \mathtt{p2} = \frac{1 - (\frac{\mathtt{q}}{\mathtt{p}})^{\mathtt{n1}}}{1 - (\frac{\mathtt{q}}{\mathtt{p}})^{\mathtt{n1+n2}}}.$$

Write a program called `gambler.py` that receives `n1` (int), `n2` (int), and `p` (float) as command-line inputs, and writes the probabilities `p1` and `p2` (separated by a space) as standard output.

```
× ~/workspace/straightline_programs
$ python3 gambler.py 10 100 0.51
0.6661883734200654 0.3338116265799349
$ python3 gambler.py 100 10 0.51
0.006110712510580903 0.9938892874894192
```

Problem 9. (*Waiting Time*) If `l` is the average number of events per unit of time, the probability `p` that one has to wait longer than time `t` until the next event is given by the exponential distribution

$$\mathtt{p} = e^{-\mathtt{lt}}.$$

Write a program called `waiting_time.py` that receives `l` (float) and `t` (float) as command-line inputs, and writes the probability `p` as standard output.

```
× ~/workspace/straightline_programs
$ python3 waiting_time.py 0.1 5
0.6065306597126334
$ python3 waiting_time.py 0.6 3
0.16529888822158656
```

Problem 10. (*Cartesian Coordinates*) Write a program called `cartesian.py` that receives `r` (float) and `t` (float) representing the coordinates of a point in polar form, converts the coordinates into Cartesian form `x` and `y` using formulae $\mathtt{x} = \mathtt{r}\cos(\mathtt{t})$ and $\mathtt{y} = \mathtt{r}\sin(\mathtt{t})$, and writes those values (separated by a space) as standard output.

```
× ~/workspace/straightline_programs
$ python3 cartesian.py 1 45
0.7071067811865476 0.7071067811865475
$ python3 cartesian.py 1 60
0.5000000000000001 0.8660254037844386
```

Problem 11. (*Great Circle Distance*) Write a program called `great_circle.py` that receives `x1` (float), `y1` (float), `x2` (float), and `y2` (float) as command-line inputs, representing the latitude and longitude in degrees of two points on Earth, and writes as standard output the great circle distance `d` (in km) between them, computed as

$$\mathtt{d} = 6359.83 \arccos(\sin(\mathtt{x1})\sin(\mathtt{x2}) + \cos(\mathtt{x1})\cos(\mathtt{x2})\cos(\mathtt{y1} - \mathtt{y2})).$$

```
× ~/workspace/straightline_programs
$ python3 great_circle.py 48.87 -2.33 37.8 -122.4
8701.387455462233
$ python3 great_circle.py 46.36 -71.06 39.90 116.41
10376.503884802196
```

Problem 12. (*Snell's Law*) Snell's law states that given two mediums, the ratio of the sines of the angles (in degrees) of incidence and refraction is equivalent to the reciprocal of the ratio of the indices of refraction of the two mediums, ie,

$$\frac{\sin(\mathtt{t1})}{\sin(\mathtt{t2})} = \frac{\mathtt{n2}}{\mathtt{n1}}.$$

Write a program called `snell.py` that receives `t1` (float), `n1` (float), and `n2` (float) as command-line inputs, and writes as standard output the corresponding angle of refraction `t2` in degrees.

```
× ~/workspace/straightline_programs
$ python3 snell.py 58 1 1.52
33.912513998258994
$ python3 snell.py 30 1 1.2
24.624318352164074
```

Problem 13. (*Uniform Random Numbers*) Write a program called `stats.py` that receives `a` (int) and `b` (int) as command-line inputs, generates three random floats (`x1`, `x2`, and `x3`), each from the interval $[\mathtt{a}, \mathtt{b})$, computes their mean $\mathtt{m} = (\mathtt{x1}+\mathtt{x2}+\mathtt{x3})/3$, variance $\mathtt{var} = ((\mathtt{x1}-\mathtt{m})^2 + (\mathtt{x2}-\mathtt{m})^2 + (\mathtt{x3}-\mathtt{m})^2)/3$, and standard deviation $\mathtt{std} = \sqrt{\mathtt{var}}$, and writes those values (separated by a space) as standard output.

```
× ~/workspace/straightline_programs
$ python3 stats.py 0 1
0.5731084550427492 0.04897843881307027 0.22131072909615176
$ python3 stats.py 50 100
91.3736830296877 25.288830238538182 5.028800079396494
```

Problem 14. (*Die Roll*) Write a program called `die_roll.py` that receives `n` (int) as command-line input, representing the number of sides of a fair die, rolls an `n`-sided die twice, and writes as standard output the sum of the numbers rolled.

```
× ~/workspace/straightline_programs
$ python3 die_roll.py 6
12
$ python3 die_roll.py 6
10
```

Problem 15. (*Triangle Inequality*) Write a program called `triangle.py` that receives `x` (int), `y` (int), and `z` (int) as command-line inputs, and writes `True` as standard output if each one of them is less than or equal to the sum of the other two, and `False` otherwise.

```
× ~/workspace/straightline_programs
$ python3 triangle.py 3 3 3
True
$ python3 triangle.py 2 4 7
False
```

Files to Submit:

1. `name_age.py`
2. `greet_three.py`
3. `day_of_week.py`
4. `three_sort.py`
5. `bmi.py`

6. `wind_chill.py`
7. `gravitational_force.py`
8. `gambler.py`
9. `waiting_time.py`
10. `cartesian.py`
11. `great_circle.py`
12. `snell.py`
13. `stats.py`
14. `die_roll.py`
15. `triangle.py`
16. `notes.txt`