Problem 1. (*Birthday Problem*) Suppose that people enter an empty room until a pair of people share a birthday. On average, how many people will have to enter before there is a match? Write a program called `birthday.py` that accepts *trials* (int) as command-line argument, runs *trials* experiments to estimate this quantity — each experiment involves sampling individuals until a pair of them share a birthday, and writes the value to standard output.

```
>_ ~/workspace/module3/assignment3

$ python3 birthday.py 1000
24
$ python3 birthday.py 1000
25
```

Directions:

- Set *count* (total number of individuals sampled across *trials* number of experiments) to 0.
- Repeat for each $t \in [1, trials]$:
  - Setup a 1D list *seen* of `DAYS_PER_YEAR` booleans, all set to `False` by default. This list will keep track of the birthdays encountered in this experiment.
  - Repeat until match:
    - Increment *count* by 1.
    - Set *birthday* to a random integer from $[0, \texttt{DAYS\_PER\_YEAR} - 1]$.
    - If *birthday* has been encountered (consult *seen*), abort this experiment, ie, break.
    - Otherwise, record the fact that we are seeing this birthday for the first time (update *seen*).
- Write the average number of people that must be sampled before a match, as an int.

Problem 2. (*Pascal's Triangle*) Pascal's triangle $\mathcal{P}_n$ is a triangular array with $n+1$ rows, each listing the coefficients of the binomial expansion $(x+y)^i$, where $0 \le i \le n$. For example, $\mathcal{P}_4$ is the triangular array:

$$\begin{array}{lllll} 1 & & & & \\ 1 & 1 & & & \\ 1 & 2 & 1 & & \\ 1 & 3 & 3 & 1 & \\ 1 & 4 & 6 & 4 & 1 \end{array}$$

The term $\mathcal{P}_n(i,j)$ is calculated as $\mathcal{P}_n(i-1,j-1) + \mathcal{P}_n(i-1,j)$, where $0 \le i \le n$ and $1 \le j < i$, with $\mathcal{P}_n(i,0) = \mathcal{P}_n(i,i) = 1$ for all i. Write a program called `pascal.py` that accepts n (int) as command-line argument, and writes $\mathcal{P}_n$ to standard output.

```
>_ ~/workspace/module3/assignment3

$ python3 pascal.py 3
1
1 1
1 2 1
1 3 3 1
$ python3 pascal.py 10
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
1 9 36 84 126 126 84 36 9 1
1 10 45 120 210 252 210 120 45 10 1
```

Directions:

- Repeat for each $i \in [0, n]$:

  - Repeat for each $j \in [1, i]$:
    - Set $a[i][j]$ to $a[i-1][j-1] + a[i-1][j]$.
- Repeat for each $i \in [0, n]$:
  - Repeat for each $j \in [0, i]$:
    - Write $a[i][j]$ followed by a space.
  - Write a newline character.

Problem 3. (*Reverse*) Write a program called `reverse.py` that accepts strings from command-line, and writes them in reverse order to standard output.

```
>_ ~/workspace/module3/assignment3

$ python3 reverse.py b o l t o n
n o t l o b
$ python3 reverse.py m a d a m
madam
```

Directions:

- Repeat for each $i \in [0, n/2]$, where n is the number of elments in a:
  - Exchange the element at i in a with the element at $n - i - 1$.
- Repeat for each $v \in a$:
  - Write v followed by a space.
- Write a newline character.

Problem 4. (*Euclidean Distance*) Write a program called `distance.py` that accepts n (int) as command-line argument, then n floats from command-line into a list x, then n floats from command-line into a list y, and writes to standard output the Euclidean distance between two vectors represented by x and y. The Euclidean distance is calculated as the square root of the sums of the squares of the differences between the corresponding entries.

```
>_ ~/workspace/module3/assignment3

$ python3 distance.py 2 1 0 0 1
1.4142135623730951
$ python3 distance.py 5 -9 1 10 -1 1 -5 9 6 7 4
13.0
```

Directions:

- Set *distance* to 0.
- Repeat for each $i \in [0, n-1]$:
  - Add square of $x[i] - y[i]$ to *distance*.
- Update *distance* to its square root.
- Write *distance*.

Problem 5. (*Transpose*) Write a program called `transpose.py` that accepts m (int) and n (int) as command-line arguments, then $m \times n$ floats from command-line into an $m \times n$ list a, and writes to standard output the transpose of a.

```
>_ ~/workspace/module3/assignment3
$ python3 transpose.py 2 2 1 2 3 4
1.0 3.0
2.0 4.0
$ python3 transpose.py 2 3 1 2 3 4 5 6
1.0 4.0
2.0 5.0
3.0 6.0
```

Directions:

- Set c (the transpose of a) to a 2D list with n rows and m columns, with all the elements set to 0.0.
- Repeat for each $i \in [0, n-1]$:
  - Repeat for each $j \in [0, m-1]$:
    - Set $c[i][j]$ to $a[j][i]$.
- Repeat for each $i \in [0, n-1]$:
  - Repeat for each $j \in [0, m-1]$:
    - Write $a[i][j]$ followed by a space.
  - Write a newline character.

Files to Submit

1. `birthday.py`
2. `pascal.py`
3. `reverse.py`
4. `distance.py`
5. `transpose.py`