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 called `wind_chill.py` that accepts t (float) and v (float) as command-line arguments, and writes the wind chill w to standard output. Your program should report the message "Value of t must be $\leq$ 50 F" if $t > 50$, and the message "Value of v must be > 3 mph" if $v \leq 3$.

```
>_ ~/workspace/module2/assignment2
$ python3 wind_chill.py 55 15
Value of t must be <= 50 F
$ python3 wind_chill.py 32 15
21.588988890532022
```

Directions:

- Use an if statement to decide when to report the error messages and when to compute and write the wind chill w.

Problem 2. (*Day of the Week*) Write a program called `day_of_week.py` that accepts m (int), d (int), and y (int) as command-line arguments, computes the day of the week (0 for Sunday, 1 for Monday, and so on) *dow* using the formulae below, and writes the day as a string ("Sunday", "Monday", and so on) to standard output.

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

```
>_ ~/workspace/module2/assignment2
$ python3 day_of_week.py 3 14 1879
Friday
$ python3 day_of_week.py 4 12 1882
Wednesday
```

Directions:

- After computing *dow*, use an if statement to write the correct output based on the value of *dow*.

Problem 3. (*Playing Card*) Write a program called `card.py` that simulates the selection of a random card from a standard deck of 52 playing cards, and writes it to standard output.

```
>_ ~/workspace/module2/assignment2
$ python3 card.py
3 of Clubs
$ python3 card.py
Ace of Spades
```

Directions:

- Set *rank* to a random integer from $[2, 14]$.
- Use an if statement to set *rankStr* to a string corresponding to *rank* — the ranks are 2, 3, ..., *Jack*, *Queen*, *King*, and *Ace*.
- Set *suit* to a random integer from $[1, 4]$.

- Use an if statement to set *suitStr* to a string corresponding to *suit* — the suits are *Clubs*, *Diamonds*, *Hearts*, and *Spades*.
- Write the desired output.

Problem 4. Write a program called `factorial.py` that accepts n (int) as command-line argument, and writes to standard output the value of $n!$, which is defined as $n! = 1 \times 2 \times \ldots (n-1) \times n$. Note that $0! = 1$.

```
>_ ~/workspace/module2/assignment2
$ python3 factorial.py 0
1
$ python3 factorial.py 5
120
```

Directions:

- Set *result* to 1.
- Repeat for each $i \in [2, n]$:
  - Update *result* to its current value times i.
- Write *result* ($n!$).

Problem 5. (*Primality Test*) Write a program called `primality_test.py` that accepts n (int) as command-line argument, and writes to standard output if n is a prime number or not.

```
>_ ~/workspace/module2/assignment2
$ python3 primality_test.py 31
True
$ python3 primality_test.py 42
False
```

Directions:

- Set i to 2.
- Repeat as long as $i \leq n/i$:
  - If i divides n, break (n is not a prime).
  - Otherwise, increment i by 1.
- If $i > n/i$, write `True` (n is a prime).
- Otherwise, write `False` (n is not a prime).

Problem 6. (*Counting Primes*) Write a program called `prime_counter.py` that accepts n (int) as command-line argument, and writes to standard output the number of primes less than or equal to n.

```
>_ ~/workspace/module2/assignment2
$ python3 prime_counter.py 10
4
$ python3 prime_counter.py 100
25
```

Directions:

- Set *count* to 0.
- Repeat for each $i \in [2, n]$:

  - Set j (potential divisor of i) to 2.
  - Repeat as long as $j \leq i/j$:
    * If j divides i, break (i is not a prime).
    * Otherwise, increment j by 1.
  - If $j > i/j$, increment *count* by 1 (i is a prime).
- Write *count* (number of primes $\leq n$).

Problem 7. (*Greatest Common Divisor*) Write a program called `gcd.py` that accepts p (int) and q (int) as command-line arguments, and writes to standard output the greatest common divisor (gcd) of p and q.

```
>_ ~/workspace/module2/assignment2
$ python3 gcd.py 408 1440
24
$ python3 gcd.py 21 22
1
```

Directions:

- Repeat as long as $p \bmod q \neq 0$:
  - Exchange p and q with q and $p \bmod q$.
- Write q (the gcd).

Problem 8. (*Sum of Powers*) Write a program called `sum_of_powers.py` that accepts n (int) and k (int) as command-line arguments, and writes to standard output the sum $1^k + 2^k + \cdots + n^k$.

```
>_ ~/workspace/module2/assignment2
$ python3 sum_of_powers.py 15 1
120
$ python3 sum_of_powers.py 10 3
3025
```

Directions:

- Set *total* to 0.
- Repeat for each $i \in [1, n]$:
  - Increment *total* by i^k.
- Write *total* (sum of powers).

Files to Submit

1. `wind_chill.py`
2. `day_of_week.py`
3. `card.py`
4. `factorial.py`
5. `primality_test.py`
6. `prime_counter.py`
7. `gcd.py`
8. `sum_of_powers.py`