Math 10 - Cumulative Project I - Introduction to Algorithms

This project will focus on the competencies *problem solving*, *technology*, and *communication*. The goal of the project is designing algorithms/programs/general solutions that could be used to complete each task. Some tasks are significantly more challenging than others.

For more information on Python, check out the website https://www.python.org.

Tasks - Measurement

Volume Computation (Required)

An engineer is designing a pool. If the pool has a length l, width w, and depth d:

i. Write a general solution for volume v in terms of l, w, and d.
Solution: $v = l \cdot w \cdot d$.

ii. Write a Python program that takes the dimensions of the pool in **feet** and calculates the volume in m^3, ft^3, and in^3.
Possible Solution:

```python
print("Given length, width, depth in feet, compute volume in ft^3, m^3, and in^3.")

l = float(input("Input length (ft): "))
w = float(input("Input width (ft): "))
d = float(input("Input depth (ft): "))

v_ft3 = l * w * d
v_m3  = v_ft3 * 0.028316846592  # 1 ft^3 = 0.028316846592 m^3
v_in3 = v_ft3 * 1728            # 1 ft^3 = 12^3 in^3

print("Volume:")
print("  ft^3:", v_ft3)
print("  m^3 :", v_m3)
print("  in^3:", v_in3)
```

Lighthouse Problem

A lighthouse is being built so that it can spread its light over an area of a m^2, where $a \in \mathbb{Q}$ and $a > 0$. What should the engineers make the height h in **meters** if the light is to reach/cover an area of a m^2, and the maximum distance a beam of light can travel is l km, where $l \in \mathbb{Q}$ and $l > 0$?

i. Write a general solution for h in terms of a and l.
Solution: Let $R = \sqrt{a/\pi}$ (radius in m) and $L = 1000\,l$ (max beam in m). Then

$$h = \sqrt{L^2 - R^2} = \sqrt{(1000\,l)^2 - \frac{a}{\pi}}.$$

ii. Write a Python program that calculates h, given a and l.
Hint: You will need to import `math` to use `math.pi`.
Possible Solution:

```python
import math
print("Height of a lighthouse given covered area (m^2) and max light distance (km).")

a = float(input("Input area a (m^2): "))
l_km = float(input("Input maximum distance (km): "))
L = l_km * 1000.0                    # km -> m
R = math.sqrt(a / math.pi)

if L < R:
    print("Max distance is too short to cover the area.")
else:
    h = math.sqrt(L*L - R*R)
    print("Required height (m):", h)
```

Savings Problem

You are saving money to purchase an item. The item costs a dollars, where $a \in \mathbb{Q}$, $a > 0$. In your bank account you have s dollars, where $s \in \mathbb{Q}$, $s > 0$. At your current job you are making m dollars a month, where $m \in \mathbb{Q}$, $m > 0$.

i. Write a general solution for l, the length in **years** it will take before you are able to purchase the item, in terms of a, s, and m.
   Solution: $l = \dfrac{a-s}{12m}$.

ii. Write a Python program that takes a, s, and m, and outputs the time (years) until you can purchase the item.
   Possible Solution:

```python
print("Time to afford an item given price a, savings s, and monthly income m.")

a = float(input("Item cost (a): "))
s = float(input("Current savings (s): "))
m = float(input("Monthly income (m): "))

if a <= s:
    print("You can already afford the item (0 years).")
else:
    t_years = (a - s) / (12.0 * m)
    print("Time needed (years):", t_years)
```

Converting Between Grams and Moles

In science class, you have been determining the molar mass of different compounds. Given a mass g in grams ($g \in \mathbb{Q}, g > 0$) and molar mass M in g/mol, convert to moles n.

i. **Solution:** $n = \dfrac{g}{M}$.

ii. **Possible Python Solution:**

```python
print("Convert grams to moles given molar mass M (g/mol).")

M = float(input("Molar mass M (g/mol): "))
g = float(input("Mass g (grams): "))

n = g / M
print("Amount (moles):", n)
```

Kinetic Energy of a Moving Object

The kinetic energy of a moving object is $E_k = \frac{1}{2}mv^2$. Energy is measured in joules:

$$1 \text{ joule} = 1 \text{ kg} \cdot \left(\frac{\text{m}}{\text{s}}\right)^2.$$

i. Write a general solution for v in **km/h** given mass m (kg) and energy E_k (J).
   Solution: $v_{\text{m/s}} = \sqrt{\frac{2E_k}{m}}, \quad v_{\text{km/h}} = 3.6\sqrt{\frac{2E_k}{m}}$.

ii. **Possible Python Solution:**

```
print("Velocity (km/h) given mass (kg) and kinetic energy (J).")

Ek = float(input("Kinetic energy Ek (J): "))
m  = float(input("Mass m (kg): "))

v_ms = (2.0 * Ek / m) ** 0.5
v_kmh = 3.6 * v_ms
print("Velocity:", v_kmh, "km/h")
```

Pythagorean Theorem in 3 Dimensions

You know the **Pythagorean Theorem** $a^2 + b^2 = c^2$. In 3D ($\mathbb{R}^3$), the length of a vector (x, y, z) is

$$\ell = \sqrt{x^2 + y^2 + z^2}.$$

i. **Solution:** $\ell = \sqrt{x^2 + y^2 + z^2}$.

ii. **Possible Python Solution:**

```
print("Length of a 3D vector given x, y, z.")

x = float(input("x: "))
y = float(input("y: "))
z = float(input("z: "))

l = (x*x + y*y + z*z) ** 0.5
print("Length:", l)
```

iii. **Challenge:** Extend to $\mathbb{R}^n$: $\ell = \sqrt{\sum_{i=1}^{n} x_i^2}$.

Tasks - Algebra and Number

Planetary Alignment Problem for Unknown Galaxy

In a solar system there are n planets ($n \in \mathbb{N}$). Each planet has a unique orbital period T_i ($i = 1, 2, \ldots, n$). If the planets are aligned at time t_0, determine the next alignment time t.

i. **Solution idea:** The next alignment occurs after the least common multiple (LCM) of the orbital periods.

ii. **Sample Python (for $n = 2$):**

```python
import math
print("Next alignment time for two planets (years).")

T1 = int(input("Orbital period of first planet (years): "))
T2 = int(input("Orbital period of second planet (years): "))

def lcm(a, b):
    return abs(a*b) // math.gcd(a, b)

print("Next alignment in", lcm(T1, T2), "years.")
```

iii. **Challenge I:** Write your own `gcd` function.

iv. **Challenge II:** Generalize to n planets.

The Locker Problem

In a school, n students are assigned n lockers. Student 1 opens all lockers; student 2 toggles lockers that are multiples of 2; in general, student i toggles lockers that are multiples of i.

i. **Which lockers stay open?** Exactly the perfect squares.

ii. **Sample Python (direct perfect-square logic):**

```python
print("Open lockers after all students pass (perfect squares).")

n = int(input("Number of students/lockers: "))
print("Open lockers:")
for i in range(1, n+1):
    if int(i**0.5) ** 2 == i:
        print(i)
```

iii. **Sample Python (simulate toggling):**

```python
print("Simulate locker toggling.")

n = int(input("Number of students/lockers: "))
open_lockers = {i: False for i in range(1, n+1)}

for student in range(1, n+1):
    for locker in range(student, n+1, student):
        open_lockers[locker] = not open_lockers[locker]

print("Open lockers:")
for i in range(1, n+1):
    if open_lockers[i]:
        print(i)
```

Iterative Approximation to a Radical

We can approximate $\sqrt{a}$ using the *Babylonian Method*:

$$x_{k+1} = \frac{x_k + \frac{a}{x_k}}{2}.$$

Example for $\sqrt{5}$ (starting at $x_0 = 2.5$) quickly converges to 2.236068.

i. **Possible Python Solution (to 10 decimal places):**

```
print("Approximate sqrt(a) to ~10 decimal places using the Babylonian method.")

a = float(input("Input a (>0): "))
x = float(input("Initial guess (>0): "))

prev = 0.0
while abs(x - prev) > 1e-10:
    prev = x
    x = 0.5 * (x + a / x)
    print(x)  # iteration trace

print("Approximation:", x)
```

ii. **Research:** Find and briefly describe another square-root algorithm.

What is the Probability of Winning?

A game has probability p $(0 \leq p \leq 1)$ of winning each round.

i. **Probability of losing a round:** $1 - p$.

ii. **Only win on the n^{th} round:** $(1-p)^{n-1}p$ (geometric distribution).

iii. **Sample Python:**

```
print("Probability of winning only on the n-th round (geometric).")

n = int(input("Number of rounds n: "))
p = float(input("Probability of win per round p (0..1): "))

q = 1.0 - p
prob = (q ** (n - 1)) * p
print("P(win only on round n) =", prob)
```

iv. **Challenge:** Probability of winning exactly once in n rounds: $\binom{n}{1}p(1-p)^{n-1} = n\,p(1-p)^{n-1}$.

Fission in a Nuclear Reactor

For a brief history on nuclear energy: https://www.youtube.com/watch?v=rcOFV4y5z8c.
In a nuclear fission reaction, each fission produces n new neutrons that can cause further fissions. After g generations, the number of fissions grows geometrically.

i. **General formula (idealized chain):** $n_{\text{fissions}} = n^g$.

ii. **Possible Python:**

```
print("Number of fissions after g generations with factor n.")

n = int(input("Neutrons produced per fission (n): "))
g = int(input("Number of generations (g): "))

f = n ** g
print("Number of fissions:", f)
```

Magical Trap in Harry Potter

In *Harry Potter and the Deathly Hallows Part II*, the Gemino curse duplicates touched objects. After touching a cup, it multiplies to become n cups per division.

i. A cup occupies v_{cup} m^3. After each division, the number of cups multiplies by n. Write the occupied volume after d divisions.
Solution: $v_{\text{occupied}} = v_{\text{cup}}\, n^d$.

ii. **Possible Python:**

```
print("Volume occupied after d divisions with factor n per division.")

n = int(input("Division factor n: "))
d = int(input("Number of divisions d: "))
v_cup = float(input("Volume of one cup (m^3): "))

v = v_cup * (n ** d)
print("Occupied volume (m^3):", v)
```

iii. **Challenge:** If room volume is v_{room} m^3, time (divisions) to fill:

$$d = \log_n\left(\frac{v_{\text{room}}}{v_{\text{cup}}}\right).$$