Mission Flatland Central Bank!

Story, Problem Tasks, and Sample Solutions: Md Faiyaz Siddiquee

The banking system in Flatland has gone rogue, and you have decided to strike back. You plan is to enter into Flatland Central Bank's main vault, which can be represented by the following figure:

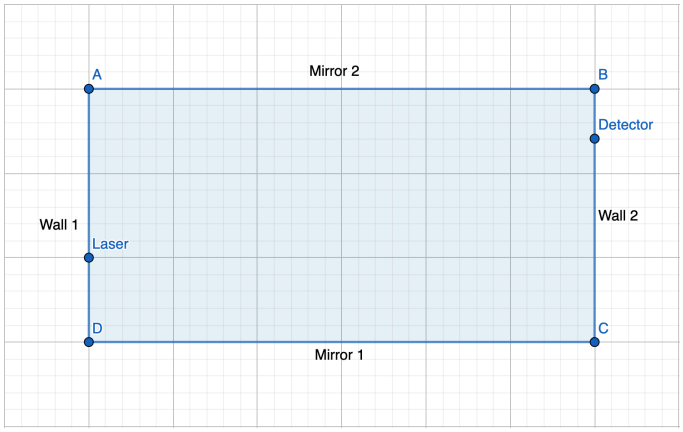

Figure 1

The vault consists of two flat walls and mirrors, arranged as a rectangle.

On Wall 1, there is a Laser placed at an arbitrary distance from point D.

On Wall 2, there is a Detector, placed at an arbitrary distance from point C, which can detect beams from the Laser.

Beyond Wall 2, there lies the secret prime number that the bank's cryptography system is based on. But in order to access that, a laser beam has to be shot (at some angle to the horizontal) from the *Laser*. The laser beam reflects off the two mirrors (following the law of reflection; angle of incidence = angle of reflection) until it reaches some point on Wall 2. If that point happens to be the *Detector*, the prime number can be accessed.

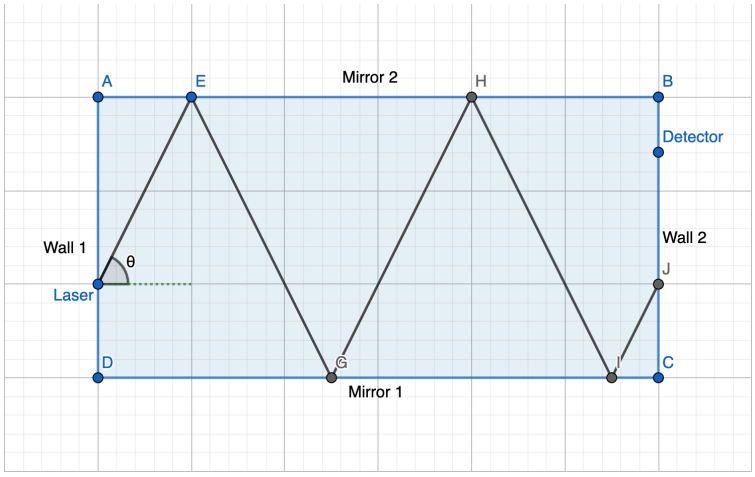

Figure 2

In the following scenario, the beam of light (path taken: Laser-E-G-H-I-J) does NOT hit the detector.

Task 1: Write a program to find out the distance between C (Refer to Figure 2) and the point where the beam collides with Wall 2 (J in the figure).

Inputs:

- Length of the vault (AB in the figure)
- Width of the vault (AD in the figure)
- Distance between D and *Laser*
- Angle of launch of light beam from horizontal in degrees (θ in the figure)

Note: θ is will be greater than -90 degrees and less than 90 degrees. Negative values of θ represent the laser beam being launched below the horizontal dotted line (see Figure 2).

Output:

- Distance between C and J

*[Since you might need to use trigonometric functions in your program, round off your final result to **3 decimal places** for simplicity].*

You enter the bank's main vault and figure out the length & width of the vault. Furthermore, you determine the distance between D and *Laser,* and the distance between C and *Detector.* You experiment with different values of θ, and using your program, find the angle which makes the beam collide Wall 2 at the same height as the *Detector.*

You obtain the secret prime number, but moments later, another vault opens up in front of you. You realise that, in fact, another prime number is required to hack into the bank's server (much like the RSA crypto system!). The next vault surely contains the other prime number!

Upon entering the next vault, you realise that this vault is the exact same as the previous one, but except for one property; the *Detector* is now moving! The bank's CEO has realised your intentions by now, and has set up the next vault so that the *Detector* initially starts on point B (see Figure 2). Just as you fire the laser, the *Detector* begins to move along BC with a constant speed. When it reaches C, it changes direction instantly and continues its constant-speed motion.

Task 2: Write a program to find out the distance between the *Detector* and the light beam when it hits Wall 2 (round it to 3 decimal places).

Inputs:

Along with the inputs of Task 1, you are given:

Speed of the light beam
Speed of the detector along BC

Note: The numerical values of speeds of both the detector and light beam are given in the same units; ***Length/Time***. *Length* is the unit that was used to express the distances in the Task 1 inputs, and *Time* is an arbitrary unit.

Output:

Distance between the *Detector* and the light beam when it hits Wall 2 (rounded to 3 decimal places).

Armed with your second program, you manipulate the launch angle until you find the case where the result of your program is 0 (the moving detector and the light beam collide with each other). Sure enough, you manage to obtain the second prime number as well. But, there is a long journey ahead of you as the saviour of the society!

Sample Solution of Task 1 (Python):

```python
import math as m

def solve_1(length, width, h_launch, theta):
    global h_hit, general_dist, ini_dist, end_dist, mode, last_dir, n_bounces_after

    if theta == 0:
        return h_launch

    elif theta > 0 and theta < 90:
        mode = "upgoing"
        ini_dist = (width-h_launch)/(m.tan(m.pi/180 * abs(theta)))

    elif theta > -90 and theta < 0:
        mode = "downgoing"
        ini_dist = h_launch/(m.tan(m.pi/180 * abs(theta)))

    else:
        return None

    general_dist = width/(m.tan(m.pi/180 * abs(theta)))

    n_bounces_after = (length-ini_dist)//general_dist
    end_dist = (length-ini_dist) - (general_dist*n_bounces_after)

    if n_bounces_after % 2 == 0:
        if mode == "upgoing":
            last_dir = "downgoing"
        else:
            last_dir = "upgoing"
    else:
        if mode == "upgoing":
            last_dir = "upgoing"
        else:
            last_dir = "downgoing"

    if last_dir == "upgoing":
        h_hit = end_dist * m.tan(m.pi/180 * abs(theta))
    else:
        h_hit = width - end_dist * m.tan(m.pi/180 * abs(theta))

    return round(h_hit, 3)
```

Sample Solution of Task 2 (Python):

```python
import math as m

def solve_2(length, width, h_launch, theta, speed_beam, speed_detector):

    if mode == "upwards":
        length_start = (width-h_launch)/(m.sin(m.pi/180 * abs(theta)))
    else:
        length_start = h_launch/(m.sin(m.pi/180 * abs(theta)))

    length_general = width/(m.sin(m.pi/180 * abs(theta)))

    if last_dir == "upgoing":
        length_end = h_hit/(m.sin(m.pi/180 * abs(theta)))
    else:
        length_end = (width-h_hit)/(m.sin(m.pi/180 * abs(theta)))

    length_total = length_start + length_general*n_bounces_after + length_end
    time_taken = length_total/speed_beam

    dist_detector = speed_detector*time_taken
    n_done = dist_detector/width
    nearest_n_whole = int(n_done)

    if nearest_n_whole % 2 == 0:
        h_detector = width - (dist_detector - (nearest_n_whole * width))
    else:
        h_detector = dist_detector - (nearest_n_whole * width)

    dist_detector_and_hit = abs(h_detector-h_hit)

    return round(dist_detector_and_hit, 3)
```