Problem Number 1

Problem Statement: No accident, please

Given 3 different sets of coordinates of airports for 'N' different flights started from the same point of time and place. Draw the flight path for the individual flights such that there is no intersection of flight paths for safety and optimization. For example: Input : Flight 1: 1,1 2,2 3,3 Flight 2: 1,1 2,4 3,2 Flight 3: 1,1 4,2 3,4 Output : Draw the path of all flights in which they had travelled.

Solution:

To Solve this problem we need to check for intersection of flight paths this can be done using checking the coordinates of the flight paths here, we can visualize the flight paths using a plotting library such as matplotlib in Python

Algorithm:

Step 1: Import Necessary Libraries like matplotlib.pyplot and itertools.combinations

Step 2: Define the Input Flight Paths

Step 3: Plot the Flight Paths

Step 4: Define a function to Plot flight paths and check for intersections

Code Implementation:

```
import matplotlib.pyplot as plt

from itertools import combinations

## Define the coordinates for each flight

flight1 = [(1,1), (2,2), (3,3)]

flight2 = [(1,1), (2,4), (3,2)]

flight3 = [(1,1), (4,2), (3,4)]

flights = [flight1, flight2, flight3]

## Plot the flight paths

def plot_flight_paths(flights):

    plt.figure(figsize=(8, 6))

    colors = ['r', 'g', 'b']

    for i, flight in enumerate(flights):

        x_coords, y_coords = zip(*flight)
```

```
        plt.plot(x_coords, y_coords, marker='o', color=colors[i], label=f'Flight {i+1}')
    plt.xlabel('X Coordinate')
    plt.ylabel('Y Coordinate')
    plt.title('Flight Paths')
    plt.legend()
    plt.grid(True)
    plt.show()
## Helper function to check intersection
def do_lines_intersect(p1, p2, q1, q2):
    def ccw(A, B, C):
        return (C[1]-A[1]) * (B[0]-A[0]) > (B[1]-A[1]) * (C[0]-A[0])
    return ccw(p1, q1, q2) != ccw(p2, q1, q2) and ccw(p1, p2, q1) != ccw(p1, p2, q2)
## Check for intersections
def check_intersections(flights):
    segments = []
    for flight in flights:
        for i in range(len(flight) - 1):
            segments.append((flight[i], flight[i+1]))

    for (p1, p2), (q1, q2) in combinations(segments, 2):
        if do_lines_intersect(p1, p2, q1, q2):
            return True
    return False
## Plot flight paths and check for intersections
plot_flight_paths(flights)
intersections_exist = check_intersections(flights)
print("Does intersections exist:", intersections_exist)
```

* If you need to handle a more complex scenario with a larger number of flights, you might need to implement an algorithm to dynamically *

Sources:

1.StackOverFlow

2.Medium Article

3.Quora Article

Code Platform Used:

Jupyter Notebook

Solution Github Link:

https://github.com/sherbinsr/Fetron-Assignment

Problem 2:

Problem Statement: My Money My Shares

Ram, Sham and Rahim went for shopping apples. They bought an apple worth 100 rupees. Ram paid 50 rupees, Sham paid 30 rupees and Rahim paid 20 rupees. Each apple is tagged with its weight on it. Write a program to distribute apples such that the quantity of apples they get is in best proportionate to the amount they paid.

Solution:

To solve this problem, we solve this using a simple Java program that first collects the apple weights from the user, then distributes the apples to Ram, Sham, and Rahim based on the amount they paid

Algorithm:

Step 1: Import Necessary Libraries like ArrayList, List, Scanner

Step 2: Collect input from user

Step 3: Define the amount paid by the each person

Step 4: Calculate the expected share for each person

Step 5: Define a function for Apple Distribution

Step 6: Display the Distribution result

Code Implementation:

```
import java.util.ArrayList;
```

```
import java.util.List;
import java.util.Scanner;

public class AppleDistribution
{
	public static void main(String[] args) {
	   Scanner scanner = new Scanner(System.in);
	   List<Integer> appleWeights = new ArrayList<>();
	   System.out.println("Enter apple weight in grams (-1 to stop):");
	   while (true) {
	      int weight = scanner.nextInt();
	      if (weight == -1)
	      {
	         break;
	      }
	      appleWeights.add(weight);
	   }

	   // Define the amount paid by each person
	   int totalAmount = 100;
	   int ramShare = 50;
	   int shamShare = 30;
	   int rahimShare = 20;
```

```
        // Calculate the expected share for each person
        double ramExpectedShare = (ramShare / (double) totalAmount)
* appleWeights.stream().mapToInt(Integer::intValue).sum();
        double shamExpectedShare = (shamShare / (double)
totalAmount) *
appleWeights.stream().mapToInt(Integer::intValue).sum();
        double rahimExpectedShare = (rahimShare / (double)
totalAmount) *
appleWeights.stream().mapToInt(Integer::intValue).sum();

        List<Integer> ramApples = new ArrayList<>();
        List<Integer> shamApples = new ArrayList<>();
        List<Integer> rahimApples = new ArrayList<>();

        // Distribute the apples
        for (int weight : appleWeights)
        {
            if (ramExpectedShare > 0)
            {
                ramApples.add(weight);
                ramExpectedShare -= weight;
            } else if (shamExpectedShare > 0)
            {
                shamApples.add(weight);
                shamExpectedShare -= weight;
```

```
            }
            else
            {
                rahimApples.add(weight);
                rahimExpectedShare -= weight;
            }
        }

        // distribution result
        System.out.println("Distribution Result:");
        System.out.println("Ram: " + ramApples);
        System.out.println("Sham: " + shamApples);
        System.out.println("Rahim: " + rahimApples);



    }
}
```

Sources:

1. Quora

Code Platform Used:

Intellji Idea

Solution Github Link:

https://github.com/sherbinsr/Fetron-Assignment