# Something about us

Have you ever wondered how systems are designed to handle billions of data points? How google/facebook serves so many requests in a fraction of second while your small program takes seconds to execute. How the software works on a distributed system. Tired of hello world applications? Is parallel computing still a theory for you? Come and join us at **Fretron LLP** to work on cutting edge technologies that can scale to process millions of data points per second. Work with young, smart and open-minded people who are passionate to solve problems through technology and coding. If you are that "curious" by nature to get to the root cause of every problem you encounter and build solutions for it, welcome to our world.

# Assignment

This is a simple exercise to unlock your potential and demonstrate to us that you can build solutions.

This package has questions with a difficulty level. Feel free to use Google to search, learn and integrate. That said, tell us the Google search terms you used and the sites. In case, your found were more useful than others (they will count POSITIVELY towards your valuation) because Honesty is one of our core values.

We care more about your thought process towards solving a problem than the solution. We were okay with incomplete solutions – but you should convince us that you have spent enough time and effort in trying to ‘get the solution.’ Oh, and yes, We will provide you more points for asking us relevant questions pertaining to this problem set( we will subjectively assess the quality of questions you ask).

Simple rules:

1. Use any programming language you are comfortable with
2. Note: if you use Google, please catalogue all your search queries and the sites that you found were most relevant to the problem you were searching for. You will get more points for using Google
3. Incomplete solutions will also be graded. We are really interested in the approach you took to solving a problem.

Restrictions:

1. Do not work in a team.
2. Do not share your answers with others. We run code comparison algorithms to look at code similarity – BTW, changing, variable names still produces similar binary instruction sets.
3. If we feel that you’ve shared your answer, you will be denied an opportunity to interview with us. Remember, we value Honesty.

Sending your solutions:

Package your solutions, Google search log + sites you found useful to solve Problem zip it, and push the Code in the Github Link and share the Updated Github Link in the Google Form Submission Link shared separately.

Evaluation Criteria:

1. Email with solutions/partially completed solutions / explanations ASAP
2. Completed solutions preferred to incomplete solutions
   1. a. We also rate an incomplete solution (depends on how much you Complete) to a hard problem higher than a completed solution to an easy problem
3. Adaptation of concepts/code from internet search. Not quoting References used, will result in disqualification for interview.
4. If solution is incomplete, we infer what you tried to do based on the code, algorithm, Google search results, etc.
5. Impress us more by using the following concepts wherever possible:
   1. Common libraries across problem sets
   2. Recursion
6. Code structure & Reusability
7. App Performance

**Problem 1 : No accident please**

Given 3 different set of coordinates of airports for ‘N’ different flights started from 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 :

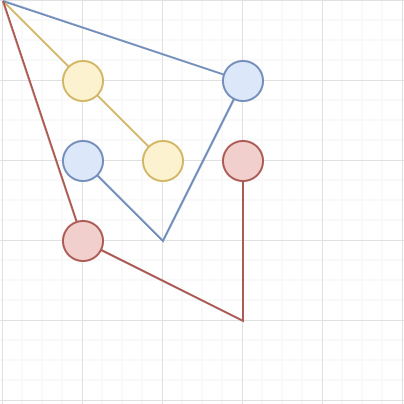
Flight 1 : 1,1 2,2 3,3

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 traveled.



**Problem 2 : My Money My Shares**

Ram , sham and rahim went for shopping apple. They bought 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 apple they get is in best proportionate to the amount they paid.

Note : you cannot cut a apple into pieces have to allocate the whole apple to one of them.

For example :

If there is 8 apple of 400g , 100g , 400g , 300g , 200g , 300g ,100g , 200g then we can distribute them as

Ram : 400g , 400g , 200g (total 1kg , 50%)

Sham : 300g , 300g (tatal 600g , 30%)

rahim , 200g , 100g , 100g (tatal 400g , 20%)

Expected program output :

run distribute\_apple

Enter apple weight in gram (-1 to stop ) : 400

Enter apple weight in gram (-1 to stop ) : 100

Enter apple weight in gram (-1 to stop ) : 400

Enter apple weight in gram (-1 to stop ) : 300

Enter apple weight in gram (-1 to stop ) : 200

Enter apple weight in gram (-1 to stop ) : 300

Enter apple weight in gram (-1 to stop ) : 100

Enter apple weight in gram (-1 to stop ) : 200

Enter apple weight in gram (-1 to stop ) : -1

Distribution Result :

Ram : 400 400 100

Sham : 300 , 300

Rahim : 200 , 100 , 100

**Problem 3 : Kill All And Return Home**

Assume you have a chessboard.

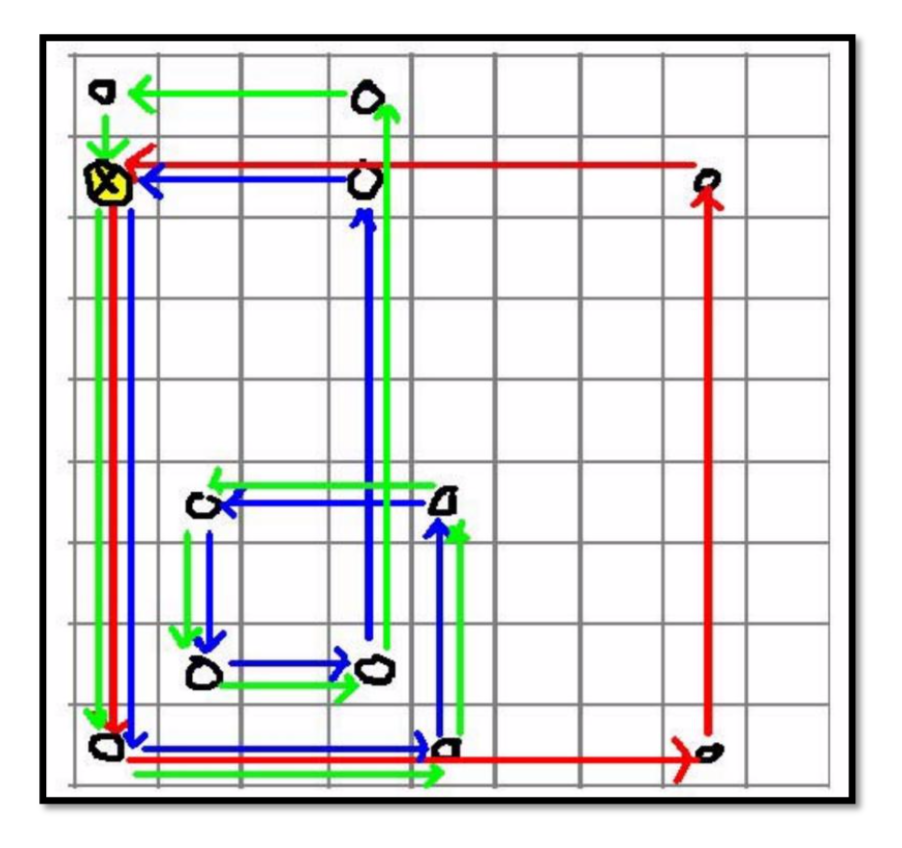
Place ‘soldiers’ on random cells on the board.

Then, place a ‘specialized castle’ in a random cell.

The properties of the ‘special castle’ are the following:

1. It can only move front.
2. When it ‘kills’ a soldier, will only move to ‘its left’
3. When it ‘kills’ a soldier, the cell becomes empty
4. Important: It can ‘jump’ over soldiers.

The following picture demonstrates 3 ways a ‘specialized castle’ can come ‘home’



Red Path:

1. Starting cell: (1,2)

2. Kill at (1,9). Then, turn left.

3. Jump over (5,9), Kill at (8,9). Then, turn left

4. Kill at (8,2). Then, turn left

5. Jump over (4,2). Return home

Green Path:

1. Starting cell: (1,2)

2. Kill at (1,9), then turn left

3. Kill at (5,9), then turn left

4. Kill at (5,6), then turn left

5. Kill at (2,6), then turn left

6. Kill at (2,8), then turn left

7. Kill at (4,8), then turn left

8. Jump over (4,2), Kill at (4,1), then turn left

9. Kill at (1,1), then turn left

10. Go home to (1,2)

Write a program for above game, sample output is given below :

Run Program :

find\_my\_home\_castle –soldiers 11

Enter coordinates for soldier 1: 1,1

Enter coordinates for soldier 2: 8,9

Enter coordinates for soldier 3: 1,9

Enter coordinates for soldier 4: 4,1

Enter coordinates for soldier 5: 4,2

Enter coordinates for soldier 6: 4,8

Enter coordinates for soldier 7: 2,6

Enter coordinates for soldier 8: 5,6

Enter coordinates for soldier 9: 8,2

Enter coordinates for soldier 10: 5,9

Enter coordinates for soldier 11: 2,8

Enter the coordinates for your “special” castle: 1,2

Thanks. There are 3 unique paths for your ‘special\_castle’

Path 1

=======

Start (1,2)

Kill (1,9). Turn Left

Jump (5,9)

Kill (8,9). Turn Left

Kill (8,2). Turn Left

Jump (4,2).

Arrive (1,2)

Path 2:

=======

Start: (1,2)

Kill (1,9). Turn Left

Kill (5,9). Turn Left

Kill (5,6). Turn Left

Kill (2,6). Turn Left

Kill (2,8). Turn Left

Kill (4,8). Turn Left Jump (4,2).

Kill (4,1). Turn Left

Kill (1,1). Turn Left

Arrive (1,2)

Path 3:

=======

Note to students: You know what Path 3 should look like.