CHESS

You are required to create a program, which simulates a **chessboard** and the movements of various types of pieces on the chessboard.

Chessboard:

The chessboard is an **8 x 8 grid** with **64 cells** in it.

With 8 rows (A, B, C…. H) and 8 columns (1, 2, 3…. 8), each cell can be uniquely identified with its **cell number**. This can be seen illustrated below.

A8	B8	C8	D8	E8	F8	G8	H8
A7	B7	C7	D7	E7	F7	G7	H7
A6	B6	C6	D6	E6	F6	G6	H6
A5	B5	C5	D5	E5	F5	G5	H5
A4	B4	C4	D4	E4	F4	G4	H4
A3	B3	C3	D3	E3	F3	G3	H3
A2	B2	C2	D2	E2	F2	G2	H2
A1	B1	C1	D1	E1	F1	G1	H1

Chess pieces and their movements:

The game of chess has **6 unique types of pieces**, with their own **unique types of movements**. These are:

1.) **King** – Can move only 1 step at a time in all 8 directions (horizontal, vertical and diagonal)
2.) **Queen** – Can move **across the board** in all 8 directions
3.) **Bishop** – Can move across the board only diagonally
4.) **Horse** – Can move across the board only in 2.5 steps (2 vertical steps and 1 horizontal step)
5.) **Rook** – Can move across the board only vertically and horizontally

6.) **Pawn** – Can move only 1 step at a time, in the forward direction, vertically. Can also move 1 step forward diagonally, in order to eliminate an opposing piece.

Objective of your program:

Your program should simulate the movement of each unique chess piece on an empty chessboard.

- **Input** – The input string to your program will be the **Type** of chess piece and its **Position** (cell number) on the chessboard. E.g. **"King D5"**
- **Output** – Once you execute the program, the output will be a string of **all possible cells in which the chess piece can move**.

Sample inputs and outputs:

Input – "King D5"

Output – "D6, E6, E5, E4, D4, C4, C5, C6"

A8	B8	C8	D8	E8	F8	G8	H8
A7	B7	C7	D7	E7	F7	G7	H7
A6	B6	**C6**	**D6**	**E6**	F6	G6	H6
A5	B5	**C5**	**D5**	**E5**	F5	G5	H5
A4	B4	**C4**	**D4**	**E4**	F4	G4	H4
A3	B3	C3	D3	E3	F3	G3	H3
A2	B2	C2	D2	E2	F2	G2	H2
A1	B1	C1	D1	E1	F1	G1	H1

Input – “Horse E3”

Output – “F5, G4, G2, F1, D1, C2, C4, D5”

A8	B8	C8	D8	E8	F8	G8	H8
A7	B7	C7	D7	E7	F7	G7	H7
A6	B6	C6	D6	E6	F6	G6	H6
A5	B5	C5	**D5**	E5	**F5**	G5	H5
A4	B4	**C4**	D4	E4	F4	**G4**	H4
A3	B3	C3	D3	**E3**	F3	G3	H3
A2	B2	**C2**	D2	E2	F2	**G2**	H2
A1	B1	C1	**D1**	E1	**F1**	G1	H1

Assumption:

Assume that the board is empty. This means that the pawn cannot move diagonally.

Our expectations:

1.) We expect you to come up with a **simple console application** in the **language of your choice**. There is no need for a UI, or a web app.

2.) We are looking for **simple, modular** and **robust design**. So focus on your best OO/ functional programming skills.

3.) Please provide ample **unit test** case coverage.

4.) Please stay within the boundaries defined in the problem. Avoid over-thinking and over-engineering.

5.) If you are using any libraries, please mention them in a README.

6.) Submit your solution in a **public Git Hub / Bit Bucket repo** with complete and atomic commit for every feature you implement. A good repo is one, which contains separate commit for each doable task/feature in the application.