Exercises 12

Task 12.1. Bulls and Cows

Create a simple single-player version of the Bulls and Cows game.

The server choses a random 4-digit secret number. All digits in the number must be different. The task of the player is to guess this number. The player provides a candidate number, and the server prints a response in form of “N bulls, M cows”. A “bull” is a correctly guessed digit in the right position. A “cow” is a digit that occurs in the secret number, but in a different position. The game continues until the player guesses the secret number.

For instance, if the secret number is “0784”, an attempt “7824” has one bull (4) and two cows (7 and 8).

Example session:

```
Server> Game start!
Player> 1234
Server> 1 bulls, 2 cows
Player> 0216
Server> 0 bulls, 2 cows
Player> 1327
Server> Yes! Game over.
```

Both a server and a client should be implemented as Akka actors, residing on the same machine (remote functionality is not necessary). Use keyboard to input player’s attempts.

Task 12.2. Online Bulls and Cows

Modify the Bulls and Cows game code to allow several remote players to play simultaneously. The server will have to keep track of connected players and provide a separate secret number to each player. The actors are supposed to live on different machines and communicate via a network in this case.

Task 12.3. Footsteps

Create an online two-player game Footsteps (also known as Tennis or Quo Vadis). The game board is a line of seven empty cells. A single token is placed into the central cell. In the beginning of the game, each player gets 50 coins. During a game turn, each player makes a secret bid by writing down a number of coins the player is willing to pay to win this turn. The bids are compared, and the winning player moves the token one cell closer to himself. If bets are equal, the token stays in place. All coins at stake are removed from the game, and the next turn begins. The game ends when some player

manages to move the token to the his closest cell and becomes a winner. If none of the players have coins left and there is still no winner, the game is declared draw.

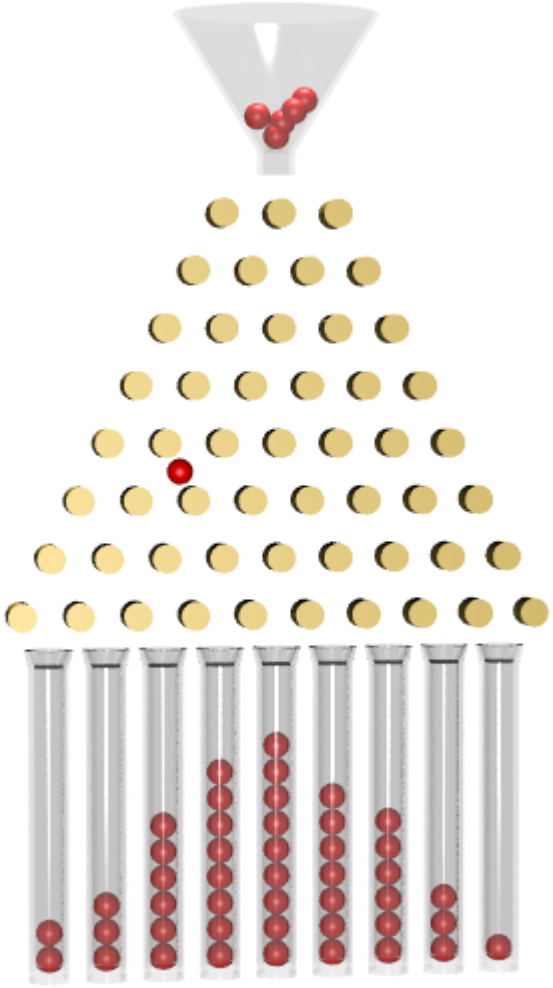
Example session:

```
Referee> Game start!
Referee> P1 (50) |.|.|.|O|.|.|.| P2 (50)

Player1> 10
Player2> 15

Referee> P1 (40) |.|.|.|.|O|.|.| P2 (35)

Player1> 5
Player2> 25

Referee> P1 (35) |.|.|.|.|.|O|.| P2 (10)
...
```

This game is supposed to be played between two player actors moderated by a referee actor. An obvious way to get started is to begin with the “Online Pig” game code discussed in lecture notes. Make sure the referee receives the messages from *both* players before proceeding to the next turn. Note that in this task you might need to use custom messages. In this case, don’t forget to make them serializable.

Task 12.4. Galton Board

The Galton board is a simple device that demonstrates the *central limit theorem* of probability theory. It consists of a vertical box with rows of pegs arranged like this:



–

Balls falling on the top peg bounce either to the left or to the right and hit some peg of the next row. Eventually each ball reaches the bottom of the device, where it is collected in a pocket. The Galton board shows that a large number of balls create a picture of a bell curve, characteristic for a *normal distribution*.

Write a program that simulates the result of *M* balls going through the Galton board with *R* rows (and *R* + 1 pockets). Speed up the process by allocating *P* Akka actors simulating their boards independently. The trick is to combine the end results of each actor when they are ready. As the final output, print ball count in each pocket.

Note: there is no “single right way” to organize this process technically. You can use pure message passing, simplify code with `ask()` or even roll out the complete `pipe()/ask()` chain.

Example input: `M = 1000000, R = 12, P = 10`

Example output: `244 2839 16126 53813 121159 193165 225405 194043 120496 53665 15883 2925 237`