**You can choose any language of his/her choice:**

Time: 1 hour

Candidates need to provide approach in first 15 min and for implementation they can use 45 minutes.

**Question 1:**

Problem Statement

* Implement the simplest version of the Snake game from the Nokia cell phones of the late 90s.
* Start with a length 5 snake at the bottom left corner and a randomly placed piece of food:

--------------------

|                  |

|                  |

|                  |

|                  |

|                  |

|                  |

|            F     |

|                  |

|SSSSS             |

--------------------

* When the player hits a key, the snake starts moving right one square per frame, which is about 500ms or so

--------------------

|                  |

|                  |

|                  |

|                  |

|                  |

|                  |

|            F     |

|                  |

|        SSSSS     |

--------------------

* Then, the player presses Up, so the snake starts going up, until it eats the food

--------------------

|                  |

|                  |

|                  |

|                  |

|                  |

|                  |

|            S     |

|            S     |

|          SSS     |

--------------------

* When the Snake hits the food, two things happen:

1. A new piece of food immediately appears in a new random location:

--------------------

|                  |

|      F           |

|                  |

|                  |

|                  |

|                  |

|            S     |

|            S     |

|          SSS     |

--------------------

2. In the **NEXT** frame, the snake grows by one:

--------------------

|                  |

|      F           |

|                  |

|                  |

|                  |

|            S     |

|            S     |

|            S     |

|          SSS     |

--------------------

We continue to eat pieces of food until eventually, we will hit the wall or our own Snake body:

1. Hit the wall:

-------------X------

|            S     |

|      F     S     |

|            S     |

|            S     |

|            S     |

|                  |

|                  |

|                  |

|                  |

--------------------

2. Hit ourselves:

--------------------

|                  |

|      F           |

|                  |

|                  |

|                  |

|          SSS     |

|          S S     |

|          SSX     |

|            SSSSS |

--------------------

**Question 2:**

Implement a parse from XML to a JS object (or Python dict, or a custom object in Java):

const xml = `

  <node>

    <hello>World</hello>

    <nested>

      <bloop>Bleep</bloop>

    </nested>

  </node>

`;

expect(parseXmlToObject(xml)).to.eq({

  node: {

    hello: 'World',

    nested: {

      bloop: 'bleep'

    }

  }

})

## **Question 3:**

Implement an in-memory file system with the following interface:

mkdir(path: string): void

writeFile(path: string, data: string): void

readFile(path: string): string

## Assumptions & Clarifications

If candidate asks clarifying questions, here are some reasonable constraints:

* Assumptions about the file system:
  + Root directory exists ie. /
* Assumptions about the path:
  + Path will always be a full path from root ie. /foo/bar
* Memory limit
  + Assume its unlimited
* Writing a file to a path that already exists
  + Overwrite the original file
* Write file/mkdir to a path where the parent directory doesn’t exist
  + Error, the full parent path must exist
* Read file from a directory
  + Error

**Question 4:**

You are given a binary tree of n nodes. While it is mostly OK, there is one extra edge that violates the tree property. Your job is to find it and eliminate it

**Question 5:**

Implement the function deepEquals which takes two arguments of any type, and returns true if they’re identical. For primitives it’s straightforward, for objects (arrays are also object), it requires a bit more.

The types we want to handle are:

* all primitives: String, Number, Boolean, Symbol, undefined
* Objects (including Arrays!)