`Functional Programming`
Project: Desert Explorer (part 1)

Assistant: Noah Van Es
`noah.van.es@vub.be`

The final grade of the course is determined by an oral exam and a project (each counts for 50% of the total grade for the course). The project consists out of two parts. A non-empty solution for both parts has to be submitted in order to get a grade. This document describes the first part of the project.

Project description

For the first programming assignment, you will have to implement an adventurer game. In this game, the player incarnates an explorer on the lookout for treasures in some desolated desert where water is scarce and lava omnipresent. His goal is to find as many treasures as possible before escaping through a portal. As he explores the map, he will have to manage his water supplies. If he runs out of water, he instantaneously dies from thirst. However, he can refill his water stock occasionally whenever he passes by an oasis. To help him in his task, he received a magic compass which indicates the distances to the closest oasis, the closest treasure and the closest portal.

Practical information

Submission The deadline for this assignment is **May 02, 2023 (23:59)**. You submit all your code and other materials in single ZIP-file through the Canvas platform.

Grading Note that your project will not only be graded on functionality, but also on the quality of your implementation (programming style, performance considerations, ...).

Individual work In case you have questions or need clarification to complete this project, feel free to contact the assistent by mail (`noah.van.es@vub.be`) or make an appointment to schedule a meeting (virtual or at my office).
The project has to be executed on a strictly individual basis! Automated tools are used to scan for plagiarism between projects; any suspicion of code exchange shall immediately be reported to the faculty's dean, and both parties shall be sanctioned in accordance with the examination rules regarding plagiarism.

Project assignment

The goal of this first programming assignment is to create a terminal-based game about exploring the desert. Your project should be entirely written in Haskell and compiled using the Glasgow Haskell Compiler[1]. A file `Main.hs` is made available on Canvas to help you with the interactive aspects of your application (that is, this file provides a game loop to read and process an input command from the terminal on each iteration).

The desert is modelled as an infinite matrix of tiles randomly generated using the function `mkStdGen`[2], using an integer seed provided by the player. The explorer starts at the position `(0,0)` and cannot walk outside of the map (negative coordinates). The explorer can move vertically (up and down) and horizontally (left and right) using the usual keys `W`, `A`, `S` and `D`. The way the map is displayed is left to your discretion. However, once a tile has been revealed by the line of sight of the explorer, it should remain so. Each move costs the explorer a measure of water and the game is lost if the player runs out of water. **Make sure that the exploration of the map does not affect its generation!** In other words, the map should be entirely defined by the player-provided seed.

There exist four different types of tiles on the map:

Desert A desert tile may contain a treasure, which should be automatically collected if the explorer walks into the tile. Otherwise, walking into a desert tile has no effect.

Water If the explorer walks (swims) into a water tile, their stock of water should automatically be refilled.

Lava If the explorer walks (jump) into a lava tile, the game is lost.

Portal If the explorer walks into a portal tile, the game is won and the sum of all collected treasures should be displayed.

The player should be able to parametrize the game to control the relative frequencies for generating these tiles. That is, the game should accept the following parameters:

s The line of sight of the explorer (as a number exploration steps).

m A number for the maximum water capacity the explorer can carry.

g The initial seed to randomly generate the map.

t The likelihood (in percentage) for a desert tile to contain a treasure.

w The likelihood (in percentage) to encounter a water tile.

p The likelihood (in percentage) to encounter a portal tile.

[1]`https://www.haskell.org/ghc/`

[2]`https://hackage.haskell.org/package/random-1.2.1.1/docs/System-Random.html#v:mkStdGen`

l The likelihood (in percentage) for lava tile generation when none of the previously-generated adjacent tiles is lava.

ll The likelihood (in percentage) for lava tile generation when at least one of the previously-generated adjacent tiles is lava.

Make sure invalid parameters are appropriately handled and do not crash the game. For instance, the application should not accept $w + p + l > 100$ nor $w + p + ll > 100$.

At any time, the player should have access to the information below:

- The total worth of collected treasures.
- The amount of water left.
- The distance to the closest water tile (as exploration steps, accounting for lava).
- The distance to the closest desert tile (as exploration steps, accounting for lava).
- The distance to the closest portal tile (as exploration steps, accounting for lava).

Provide two versions of your distance computation function. One should be lazy while the other should not contain any memory leak. Demonstrate the difference in memory consumption between both versions by tuning the game parameters so that it can be noticed by profiling the heap. Include the resulting graphs in your submission archive.