Homework 3
SNU 4910.210 Fall 2012

Chung-Kil Hur

due: 10/19(Sun) 24:00

Exercise 1 "Manual Type Checking"

Type check your solutions (or the official solutions) for homework assignments 1.3, 1.5, 1.6, 2.1, 2.2, and 2.3.

Exercise 2 "Maze Validation"

A maze is defined as follows.

- A maze consists of $n \times m$ rooms arranged in n rows and m columns.
- Two contiguous rooms may or may not have a wall between them.
- There are two special rooms: one for the start in the top row and one for the end in the bottom row.

Define a function `maze-check` that checks whether a given maze is valid or not.

$$\texttt{maze-check} : \mathit{maze} \times \mathit{room} \times \mathit{room} \to \mathit{bool}$$

More specifically, `maze-check` determines, given a maze M and two rooms r_1, r_2, whether there is a path from r_1 to r_2 in the maze M. Then, we can check the validity of any given maze by checking whether there is a path from the start room to the end one using `maze-check`.

You can define `maze-check` using the following functions without knowing how they are implemented. TAs will provide an implementation of those functions for you.

$$\begin{aligned}&\texttt{can-enter} : \mathit{maze} \times \mathit{room} \to \mathit{room\ \ list}\\ &\texttt{same-room?} : \mathit{room} \times \mathit{room} \to \mathit{bool}\\ \\ &\texttt{empty-set} : \mathit{room\ \ set}\\ &\texttt{add-element} : \mathit{room} \times \mathit{room\ \ set} \to \mathit{room\ \ set}\\ &\texttt{is-member?} : \mathit{room} \times \mathit{room\ \ set} \to \mathit{bool}\\ &\texttt{is-subset?} : \mathit{room\ \ set} \times \mathit{room\ \ set} \to \mathit{bool}\end{aligned}$$

`can-enter` returns, given a room r, the list of those rooms r' that are next to r with no wall between r and r'. `same-room?` determines whether given two rooms are the same one or not.

Also, show that your function `maze-check` always terminates. □

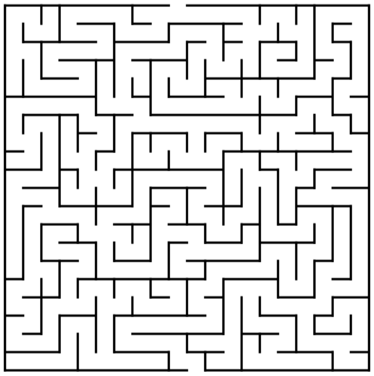
Exercise 3 "Maze Generation"

Define a function

$$\texttt{mazeGen} : \mathit{int} \times \mathit{int} \to \mathit{maze}$$

that generates, given two positive integers n and m, a valid maze of size $n \times m$ at random.

Here is an example of maze:



Use the following functions that will be provided by TAs. Your implementation of `mazeGen` should not depend on how the functions provided are implemented.

$$\begin{aligned}
&\texttt{init-maze} : \mathit{int} \times \mathit{int} \to \mathit{maze}\\
&\texttt{open-s} : \mathit{int} \times \mathit{int} \times \mathit{maze} \to \mathit{maze}\\
&\texttt{open-n} : \mathit{int} \times \mathit{int} \times \mathit{maze} \to \mathit{maze}\\
&\texttt{open-e} : \mathit{int} \times \mathit{int} \times \mathit{maze} \to \mathit{maze}\\
&\texttt{open-w} : \mathit{int} \times \mathit{int} \times \mathit{maze} \to \mathit{maze}\\
&\texttt{maze-pp} : \mathit{maze} \to \mathit{void}
\end{aligned}$$

`(init-maze` n m`)` returns the maze of size $n \times m$ that is fully blocked, meaning that it has a wall between every pair of two contiguous rooms. Each room has a coordinate (i, j) with $0 \leq i \leq n-1$ and $0 \leq j \leq m-1$. `(open-s` i j M`)` removes the south wall of the room at (i, j) from the maze M. The functions `open-n`, `open-e` and `open-w` similarly remove the north, the east and the west wall, respectively. `(maze-pp` M`)` prints the maze M.

A reasonable way to generate a valid random maze is to first randomly choose a start room in the top row and an end room in the bottom row; and then repeatedly remove a randomly chosen wall until there is a path from the start room to the end one.

You can use the function `random`: see `http://tinyurl.com/o7jvlyb` for the details. □