Principles of Programming, Fall 2009
Practice 12
Term Project Exercises

Woosuk Lee, Suwon Jang, Sungkeun Cho
Programming Research Lab.@SNU

November 30, 2009

The goal of this lab is implementing a simple shortest-path search algorithm for the term project. We're going to make a weighted directed graph from a game board first. Then find the shortest path of the graph from one location to another location.

A game board is constructed by only two kinds of tile `Plain` and `Wall`. You consume 1 time unit for moving or destructing a wall. So you need 2 time units for moving to a wall tile.

For simplicity we'll use more abstract board type instead of string. `board` type is defined as following.

```
type loc = int * int
type tile = Plain | Wall
module LocMap = Map.Make(struct type t = loc let compare = compare end)
type board = tile LocMap.t
```

`board` is a mapping from (x,y) location to a tile. (e.g. $\{(0,0) \mapsto$ `Plain`, $(1,0) \mapsto$ `Wall`, $\ldots\}$)

1. Define the converting function `makeGraph` from a `board` to a weighted directed graph. Each vertex represents a location, each edge is a way to another location and weight is the number of consuming time units. `graph` type is

   ```
   type weight = int
   type graph = ((loc * weight) list) LocMap.t
   ```

   And `makeGraph` function has a following type.

   ```
   makeGraph : board -> graph
   ```

2. Define the function `shortestPath`. *shortestPath from to g* returns shortest path from *from* to *to* in the graph *g* and the number of consuming time units. The shortest path is the least time comsuming path. `shortestPath` has a follwing type.

   ```
   shortestPath : loc -> loc -> graph -> (loc list * weight)
   ```

TA provides skeleton code and visualizer from the lecture board.