Assignment 2: The Maze

5DV086, Fall 2008

January 19, 2009

Assignment

In one respect this assignment is not straightforward: you must find a path through a maze. Your task is to search through an arbitrarily large maze from its start point to its end point. A maze is represented as a grid of walls, pathways, and a unique start and end point. A maze is stored as a list of strings with special symbols to identify the type of each square.

The symbols are:

```
"!", "+", "-"    (Wall)
" "              (Pathway)
"S"              (Start point)
"E"              (End point)
```

An example maze and its representation as a list of strings:

```
Maze             Representation
--------------------------------

+-----+          ["+-----+",
!S    !           "!S    !",
!-+ + +--+        "!-+ + +--+",
!       !         "!       !",
! +-+ +--+        "! +-+ +--+",
!    E!           "!    E!",
+-----+           "+-----+"]
```

The following assumptions apply:

- All mazes have a unique start and end point and the end is always reachable from the start.
- Not all squares are necessarily reachable from the start point.
- There is no way to escape the maze from the start point (i.e., all mazes are closed).

Implementation Requirements

The output of your maze solving function (`maze`) should be a list that enumerates all the steps required to go from the start to the end point. You must use the `Direction` datatype:

```
datatype Direction = north | south | east | west
```

The `maze` function takes a single argument which is a list-representation of a maze as discussed above. The function's output should have the type `Direction list`. Example usage:

```
- maze maze1; (* The same maze as above *)
> val it = [east, east, south, south, east, east, south, south]
: Direction list
```

Hints

- You are not required to find the shortest path. Any valid path will do.
- Use for instance a depth-first search, A*, or Dijkstra's shortest path algorithm.
- A set of example mazes for testing: `mazes.sml` (download from the course web page).

Report

The report is a major part of this assignment.

- **Presentation.** Your report should be structured, self-contained, spell checked, and nicely formatted. The title page must include your name, user name at the Computing Science department, and the name of the course (Programspråk, 5DV086, HT08).
- **Source code.** Pretty-printed source code should be attached to the report (e.g., use the `a2ps` utility). Electronic access to your code must be provided through your home directory. Make sure you specify the path in your report.
- **Content.** Since the report is supposed to be self-contained, you must include proper introduction material. Describe your solution and include details on algorithms, internal representation(s), testing, etc.

This assignment should be solved **individually** and the report must be handed in no later than **17 April, 2009, at 17:00**.