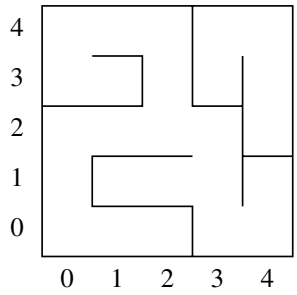
COMPUTER SCIENCE TRIPOS Part IA – 2011 – Paper 1

1 Foundations of Computer Science (MOM)

This question has been translated from Standard ML to OCaml

This question considers 2-dimensional rectangular labyrinths such as the following.



Here horizontal and vertical bars indicate *walls*. One can *step* from a position to an adjacent position if there is no wall obstructing the move. For example, one can move *in one step* from position (1, 1) to position (2, 1), but not to (1, 0). Diagonal steps are not allowed.

(*a*) Carefully explain a convenient way of representing such labyrinths in OCaml. Then define a function, call it `next`, which takes two arguments, a position `(x, y)` and a labyrinth, and returns a list of all positions that can be reached *in one step* from position `(x, y)`. [Hint: Consider representing labyrinths as functions with a type of the form `int * int ->` *something*.] [8 marks]

(*b*) Use `next` to code, in OCaml, a space-efficient function that checks whether it is possible to move from a position `(x1, y1)` to another position `(x2, y2)` in a given labyrinth. For full marks, make sure your function always terminates. [6 marks]

(*c*) Explain, not necessarily using OCaml code, how one can code a function that returns the *shortest path* between two given positions. Here path means a list of all the positions one would visit en route to the destination. The solution must be space efficient and practical even for large labyrinths. Briefly explain why your solution is space efficient. [6 marks]