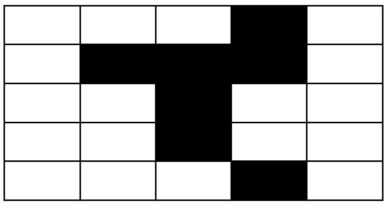
CENG 242

Homework #6

(Due: June 4th, 2010 Friday 23:55)

In this homework you are asked to find all rooms that are accessible from a given starting room in a maze. The maze is a rectangular grid of rooms, some of which are occupied, and some are empty. You are only allowed to move through empty rooms. While traversing the maze, you are only allowed to move up, down, left and right; diagonal moves are not allowed. The following is a sample maze. The black colored rooms are occupied, and white ones are empty.



The maze will be given to you as a list in a row-major order. Empty rooms will be represented with zeros and occupied rooms will be represented with ones. So the above maze will be given as;

```
[[0, 0, 0, 1, 0], [0, 1, 1, 1, 0], [0, 0, 1, 0, 0], [0, 0, 1, 0, 0],
[0, 0, 0, 1, 0]]
```

You will write a predicate called accessible/3 in the following form.

```
accessible(Maze, (StartRow, StartColumn), X).
```

Where the first parameter will be the maze, the second parameter is a tuple that represents the starting room. Room indexes start with one. So, the top left corner of the maze will be (1, 1). The final parameter will be the variable that you will unify with the rooms that are accessible from the starting room. In order to take grade, you have to generate all possible answers for this variable. For example, when we invoke your predicate with the above maze and starting room of (1, 1) your predicate should give an output similar to the following:

```
?- accessible ([[0, 0, 0, 1, 0], [0, 1, 1, 1, 0], [0, 0, 1, 0, 0],
[0, 0, 1, 0, 0], [0, 0, 0, 1, 0]], (1, 1), X).
X = (1, 1);
X = (1, 2);
X = (1, 3);
X = (2, 1);
X = (3, 1);
X = (3, 2);
X = (4, 1);
X = (4, 2);
X = (5, 1);
```

```
X = (5, 2);
X = (5, 3);
false.
```

The order of the rooms listed is not important. Note that in case a room is accessible from different paths, you may list it more than once. For this homework, this is OK. But you can get a ***bonus*** of 20 points if you don't include such duplicates in your output.

Specifications

- All the work should be done individually.
- Submit a single file called "hw6.pl" through the COW system.
- Code your programs using only SWI-Prolog or GNU Prolog and test them on inek machines before submitting.
- The evaluation will be black box. We will only look at your code if it doesn't work.
- Inputs that are too large will not be tried.