

# Programming Paradigms 2017-18

## Logic Programming with Prolog

### Purposes of this assignment

To give you experience using the logic programming paradigm and writing a “real” program in Prolog. The program should solve a simple “maze”.

### General idea

Consult a file [maze.pl](#), containing a maze. Here is a sample file:

```
mazeSize(5, 9).  
barrier(1, 8).  
barrier(2, 1).  
barrier(2, 2).  
barrier(2, 4).  
barrier(2, 5).  
barrier(3, 4).  
barrier(3, 7).  
barrier(3, 9).  
barrier(4, 4).  
barrier(4, 7).  
barrier(4, 8).  
barrier(4, 9).  
barrier(5, 2).
```

which represents the following maze:

	1	2	3	4	5	6	7	8	9
	+-----+								
1		.	.	.	.	.	.	x	
2		x	x	.	x	x	.	.	
3		.	.	.	x	.	.	x	
4		.	.	.	x	.	.	x	
5		.	x	.	.	.	.	.	
	+-----+								

Your task is to write a predicate

`solve(From, To, Path)`

which, given locations `From` and `To`, finds a `Path` going from `From` to `To`. `From` and `To` are given as two element lists, and `Path` should be a list of two-element lists. The first element of `Path` should be `From`, and the last element should be `To`. Moves can be made horizontally or vertically, but not diagonally.

For example,

`solve([3,2], [2,6], [[3,2], [3,3], [2,3], [1,3], [1,4], [1,5], [1,6], [2,6]])`.

should print the solution as a list, and also as a text drawing, for example, as

```
      1  2  3  4  5  6  7  8  9
    +-----+
1  | .  .  *  *  *  *  .  x  . |
2  | x  x  *  x  x  *  .  .  . |
3  | .  *  *  x  .  .  x  .  x |
4  | .  .  .  x  .  .  x  x  x |
5  | .  x  .  .  .  .  .  .  . |
    +-----+
```

You do not have to find the shortest path (although that would be nice), nor does your text drawing have to look exactly like our one; but it should be possible to see the path.

Please put your solution on a file named `maze-solver.pl`.

## Submission

As part of your portfolio; see the website for due dates.