Project 02

Due: Thursday, December 7th 11:59pm

1 Description

In this project you will write a prolog program that can solve a maze, creating a predicate `find_exit/2` for the user to query.

2 Details

You will write a predicate `find_exit/2`. The first parameter is the maze as described below, and the second should be a list of actions that will solve the maze. The predicate succeeds if following the list of actions will lead from the start space to any exit space. The predicate must work even when called with an unbound variable for its second parameter. The predicate should fail if the maze is invalid, if following the actions does not end in an exit space, or if an action would result in moving off the maze or onto a wall.

2.1 The Maze

A maze is a list of rows with each row being a list of cells. Each cell can be a `f`, `w`, `s`, or `e`.

- An `f` represents a floor space (empty).
- A `w` represents a wall.
- An `s` represents the start space. There should be exactly one start space.
- An `e` represents an exit. There can be more than one exit.

2.2 The Actions

An action can be `left`, `right`, `up`, and `down`. When solving the maze, the coordinates start on the start space.

- A `left` action moves the coordinates to the previous column.
- A `right` action moves the coordinates to the next column.
- A `up` action moves the coordinates to the previous row.
- A `down` action moves the coordinates to the next row.

2.3 Some Tips

- Don't forget that a predicate can have more than one rule.
- Just like when we did the nth predicate for tictactoe, many predicates can be done quite simply. Take advantage of unification, and try not to overcomplicate your solutions.
- You can use negation-as-failure and the cut to simplify some predicates
- When using the commandline, you can load multiple files. Use the included files to help with testing.
  - The included `example.pl` file contains some simple example mazes and a predicate to pretty print a maze.
  - The included `test.pl` provides `gen_map/4`. The first parameter should be an integer greater than zero. Four is a good value. It controls the complexity of the maze. The second and third parameters are the number of rows and columns (resp.) for the maze you want to generate. The fourth parameter will unify with the generated maze.

3 What to Turn in

Upload your submission as a zip archive containing the following:

- Source code (.pl file(s))
- Readme (Plain text document)
  - List the files included in the archive and their purpose
  - Explain how to compile and run your project
  - Include any other notes that the TA may need
- Write-up (Microsoft Word or pdf format)
  - How did you approach the project?
  - How did you organize the project? Why?
  - What problems did you encounter?
  - How did you fix them?
  - What did you learn doing this project?
  - If you did not complete some feature of the project, why not?
    - What unsolvable problems did you encounter?
    - How did you try to solve the problems?
    - Where do you think the solution might lay?
      - What would you do to try and solve the problem if you had more time?

4 Grading

The grade for this project will be out of 100, and broken down as follows:

Followed Specifications	50
Following the Logical Paradigm	20
Correct Output (Actions are correct, Fails when it is supposed to, etc.)	20
Write-up	10

If you were not able to complete some part of the program discussing the problem and potential solutions in the write-up will reduce the points deducted for it. For example, suppose there is a bug in your code that sometimes allows two customers to approach the same worker, and could not figure out the problem before the due date. You can write 2-3 paragraphs in the write-up to discuss this issue. Identify the error and discuss what you have done to try to fix it/find the problem point, and discuss how you would proceed if you had more time. Overall, inform me and the TA that you know the problem exists and you seriously spend time trying to fix the problem. Normally you may lose 5 points (since it is a rare error) but with the write-up you only lose 2. These points can make a large difference if the problem is affecting a larger portion of the program.