Project 4 Hints

Attached Files:

- search.pl (907 B)

Quite a few people have said they dont know where to start project 4.

As requested here is a minimal implementation of wumpus.pl
. It will lead to infinite execution but it can be used to test. It will work on a trivial map like

```
....
....
....
.W..
```

The key ideas are these:

- The state must keep track of how much of the map you have explored
- Each guess must reveal more information about the map
- Initially treat all feedback: smell, stench, damp as the same as empty just concentrate on recording the map accurately. Later you can try to improve your solution by making use of this information

Some code for finding a path in a graph is given in search.pl

- Note how it keeps track of previous places visited so it doesn't go in circles.
- It will search for all possible (simple) paths, e.g.

find(a,g,P) gives answers


```
?- find(a,g,P).
P = [east, south, south] ;
P = [east, east, south, west, south] ;
false.
```

We can use the same code for searching for all possible destinations with their path

```
?- find(a,D,P).
D = a,
P = [] ;
D = b,
P = [east] ;
D = d,
P = [east, south] ;
D = e,
P = [east, south, east] ;
D = c,
P = [east, south, east, north] ;
D = f,
P = [east, south, east, east] ;
D = i,
P = [east, south, east, east, south] ;
D = h,
P = [east, south, east, east, south, west]
...
```

So the key steps in a basic solution are:

- represent what you know about the map (what is in each square. This mainly starts as unknown)
- Use find to find a path to a square which is currently unknown
- Write updateState (rather like find) that mimics the instructions in Guess and uses the feedback to update your knowledge of the map
- When you find the wumpus (through feedback) change your guessing to
  - look for a Path to a place, such that its above the wumpus and on the same column, where you can safely move south, and where no pits are in between you and the wumpus, and make guess append(Path, [south,shoot],Guess). Of course you have to consider the cases shooting from left right and below as well. The key here is to use Prolog to search for a path to such a place.
  -

OTHER THINGS TO THINK ABOUT

Remember you will not always go where you plan to. Given the map

```
.#...
.....
.#...
```

and starting from (1,1) the guess [east, east, east, east, south, west, west, west, west, south, east, east, east, east] leads to feedback [wall,wall,wall,wall,empty,wall,wall,wall,wall,empty,wall,wall,wall,wall]. The robot has only moved directly south. The part of the map that can be updated is

```
.#
.
.#
```