جامعة الأخوين

ⵜⴰⵙⴷⴰⵡⵉⵜ ⵏ ⵓⵅⴰⵡⴰⵢⵏ

AL AKHAWAYN UNIVERSITY

Project 2: Logical Agent for Wumpus Game

Abdelghafour Abdou <86136>

Dr. Tajjeedine Rachidi

Teammate: Abderafii Abdou <86138>

Intro. to Artificial Intelligence

June 29th, 2022

I. The project objective:

The aim is to build a single iteration of a logical agent, that given (TOLD) a starting position and ASKED about one single action (safety of rooms, shooting, or picking up gold) will return a reply. That is, your agent will not try to work out the best action, but just answer questions ASKED by a human given a knowledge base (rules of the game and the starting configuration). The game is a simple world example to illustrate the worth of a knowledge-based agent and to represent knowledge representation. The game is represented with a cave that has n*n rooms that are connected with each other. One room within the cave had the Wumpus, who is a beast that kills everyone. You can kill the beast by using the only arrow that the player has. Within the journey of the agent during the cave, there are some pits that are bottomless rooms in which one will be stuck there. There is also another chamber that contains gold, that should be collected.

II. Key Predicates, and the meaning of the Variables

Key Predicates:

room(X,Y): The room at the coordinates X(Horizontally) and Y(Vertically).

Wumpus(room(X,Y)): The Wumpus is at the room with the coordinates X(Horizontally) and Y(Vertically).

Pit(room(X,Y)): The Pit is at the room with the coordinates X(Horizontally) and Y(Vertically).

wumpusisdead(0 or 1): 0 if Wumpus is alive, and 1 if Wumpus is dead.

Startingpoint((X,Y)): The starting point where the agent starts.

Gold(room(X,Y)): The location of the gold in the map with the coordinates X(Horizontally) and Y(Vertically).

breeze(room(X,Y)): creates breeze so we can know the location of pits in our game.

stench(room(X, Y)): creates stench so we can know the location of pits in our game.

adjacentto(room(X,Y)): sees which rooms are Adjacent to the player.

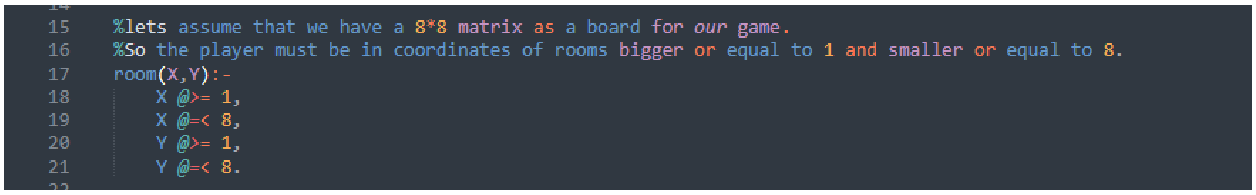
checkifsafe(room(X,Y)): check if a specific room is safe.

checkroomontheleft():/checkroomontheright():/checkroomonsouth():/checkroomonnorth(): checks if the adjacent rooms are safe or not.

grabgold(room(X1,Y1)): grab the gold if we are in the same room as the gold

shootwumpus(room(X1,Y1)): shoots the wumpus if its in one of the adjacent rooms to the player.

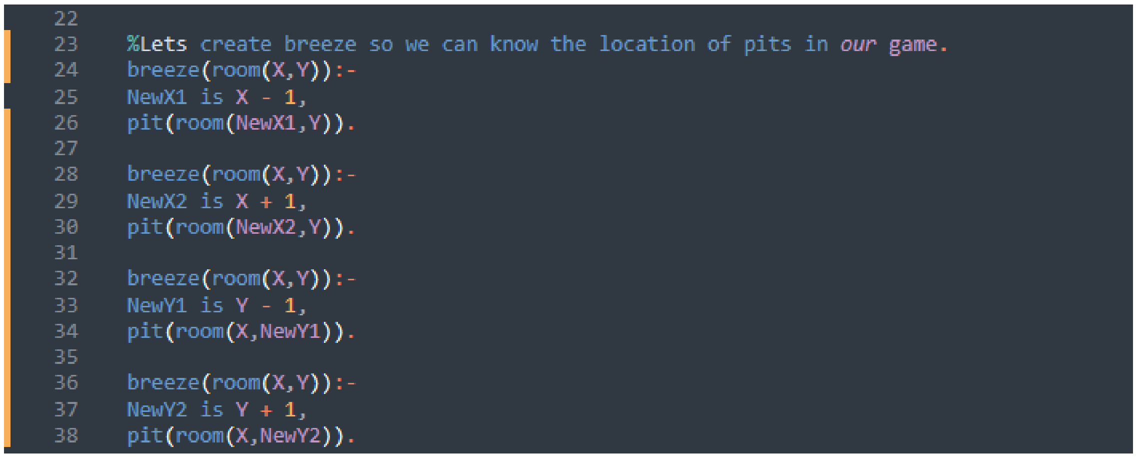
Room:



```
%lets assume that we have a 8*8 matrix as a board for our game.
%So the player must be in coordinates of rooms bigger or equal to 1 and smaller or equal to 8.
room(X,Y):-
    X @>= 1,
    X @=< 8,
    Y @>= 1,
    Y @=< 8.
```

This is a small function to make sure that if the user enters an invalid room coordinate, the code will print back false. It also gives the user and idea about the size of the matrix or the board.

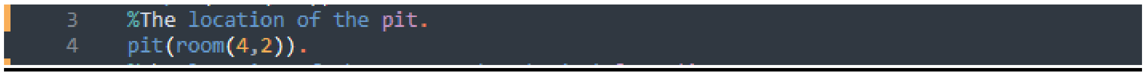
Breezes:



```
%Lets create breeze so we can know the location of pits in our game.
breeze(room(X,Y)):-
NewX1 is X - 1,
pit(room(NewX1,Y)).

breeze(room(X,Y)):-
NewX2 is X + 1,
pit(room(NewX2,Y)).

breeze(room(X,Y)):-
NewY1 is Y - 1,
pit(room(X,NewY1)).

breeze(room(X,Y)):-
NewY2 is Y + 1,
pit(room(X,NewY2)).
```

In this part, we tried to create the breeze indicator in each room adjacent to a pit by calculating the coordinate of the adjacent rooms to the pit, all the 4 possibilities of the room's coordinates are included.

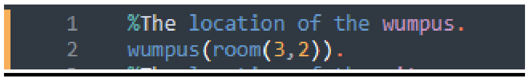
Pits:



```
%The location of the pit.
pit(room(4,2)).
```

We created a pit with already known coordinates in order to simplify the testing.

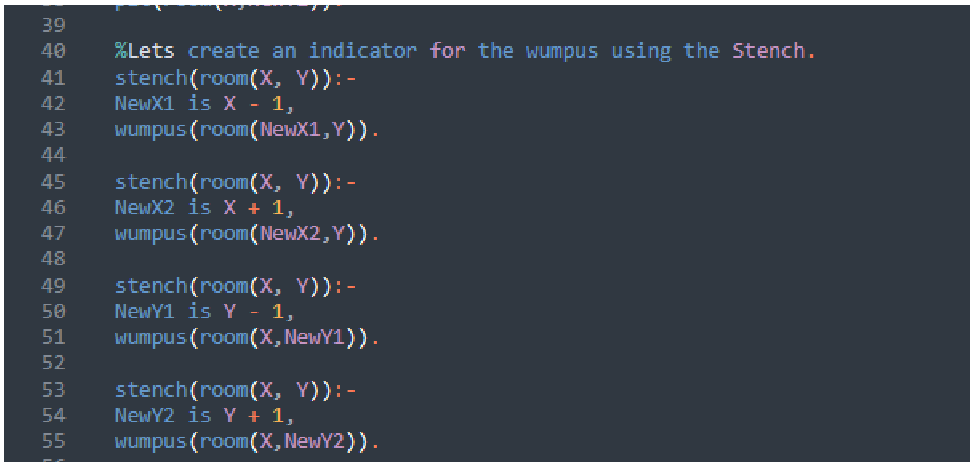
Wumpus:



```
%The location of the wumpus.
wumpus(room(3,2)).
```

We created a Wumpus with already known coordinates for the sake of making the testing easier.

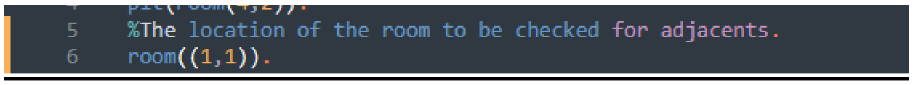
Stench:



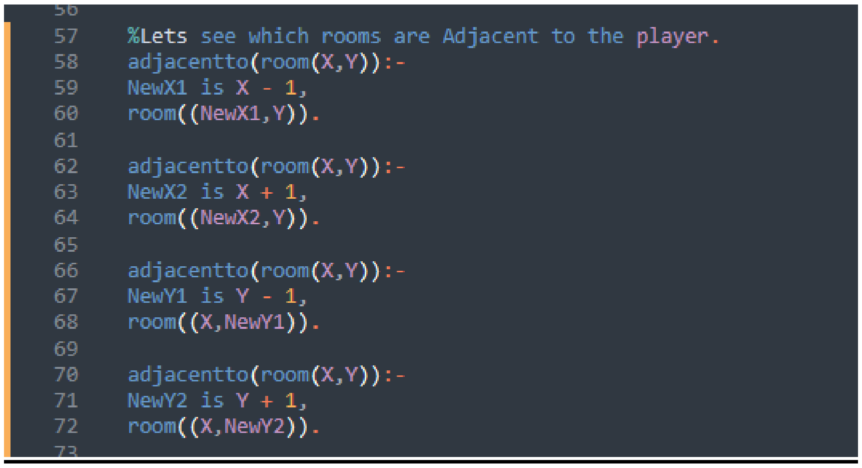
```
%Lets create an indicator for the wumpus using the Stench.
stench(room(X, Y)):-
NewX1 is X - 1,
wumpus(room(NewX1,Y)).

stench(room(X, Y)):-
NewX2 is X + 1,
wumpus(room(NewX2,Y)).

stench(room(X, Y)):-
NewY1 is Y - 1,
wumpus(room(X,NewY1)).

stench(room(X, Y)):-
NewY2 is Y + 1,
wumpus(room(X,NewY2)).
```

We created a stench coming from the Wumpus in the nearby cells by calculating the coordinate of the adjacent rooms to the Wumpus, all the 4 possibilities of the room's coordinates are included.

Adjacent To:



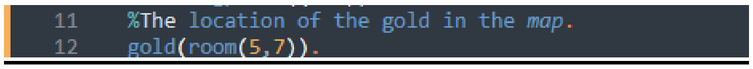
```
%The location of the room to be checked for adjacents.
room((1,1)).
```



```
%Lets see which rooms are Adjacent to the player.
adjacentto(room(X,Y)):-
NewX1 is X - 1,
room((NewX1,Y)).

adjacentto(room(X,Y)):-
NewX2 is X + 1,
room((NewX2,Y)).

adjacentto(room(X,Y)):-
NewY1 is Y - 1,
room((X,NewY1)).

adjacentto(room(X,Y)):-
NewY2 is Y + 1,
room((X,NewY2)).
```

We created a function to check if a room given by the user is adjacent to a predeclared room, in this case the predeclared room is (1,1).

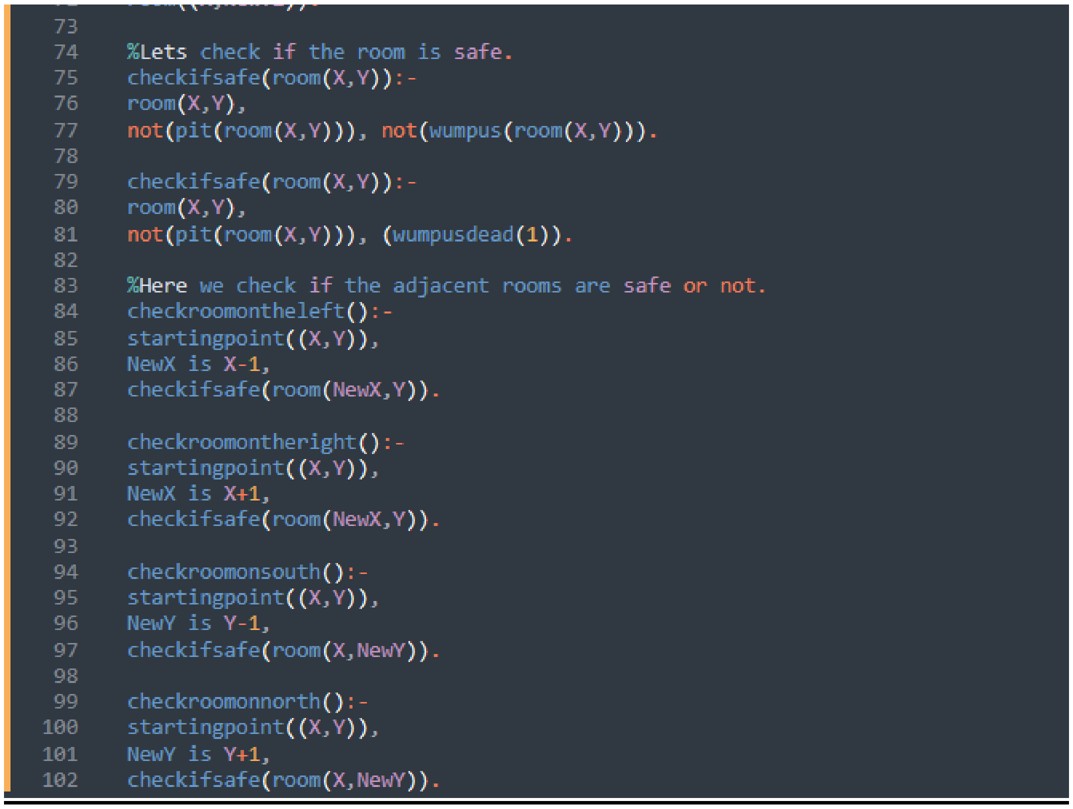
Gold:



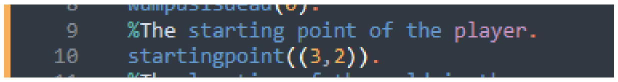
```
%The location of the gold in the map.
gold(room(5,7)).
```

We created gold in a certain room manually with already known coordinates in order to simplify the testing.

Safe:



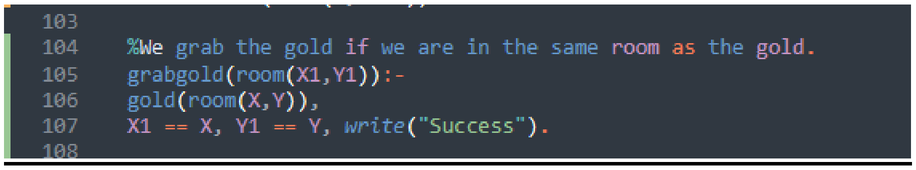
```
%Lets check if the room is safe.
checkifsafe(room(X,Y)):-
room(X,Y),
not(pit(room(X,Y))), not(wumpus(room(X,Y))).

checkifsafe(room(X,Y)):-
room(X,Y),
not(pit(room(X,Y))), (wumpusdead(1)).

%Here we check if the adjacent rooms are safe or not.
checkroomontheleft():-
startingpoint((X,Y)),
NewX is X-1,
checkifsafe(room(NewX,Y)).

checkroomontheright():-
startingpoint((X,Y)),
NewX is X+1,
checkifsafe(room(NewX,Y)).

checkroomonsouth():-
startingpoint((X,Y)),
NewY is Y-1,
checkifsafe(room(X,NewY)).

checkroomonnorth():-
startingpoint((X,Y)),
NewY is Y+1,
checkifsafe(room(X,NewY)).
```



```
%The starting point of the player.
startingpoint((3,2)).
```

In this function, we check if a room is safe, the coordinates of the room are given by the user and they are used to check if the room is safe, by checking if there is no pit in the room and if there is no Wumpus in the room or if its dead. Moreover, we can check if a room to the right, left, north or south is safe by simply giving a starting point which is already declared to make testing easier, and using the checkifsafe() function again on the visited room.

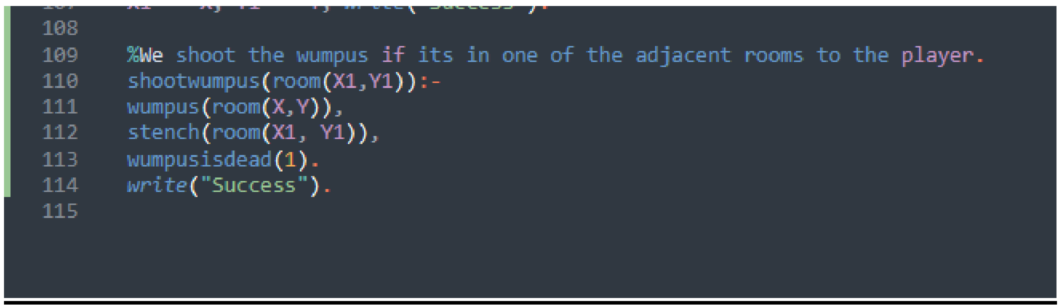
Grab gold:



```
%We grab the gold if we are in the same room as the gold.
grabgold(room(X1,Y1)):-
gold(room(X,Y)),
X1 == X, Y1 == Y, write("Success").
```

In this function, the user can pick up the gold if he is in the same room as the gold and that is achieved through a comparison between X1 and Y1 which are the coordinates of the player and X and Y which are the coordinates of the gold.

Shoot Wumpus:



```
%We shoot the wumpus if its in one of the adjacent rooms to the player.
shootwumpus(room(X1,Y1)):-
wumpus(room(X,Y)),
stench(room(X1, Y1)),
wumpusisdead(1).
write("Success").
```

The player shoots the Wumpus if he is in an adjacent room to the Wumpus, we can make sure that this information is obtained throughout the use of the function stench(room(X1,Y1)) which returns true if the room is adjacent to a Wumpus, otherwise if returns false , if the returned value is true then we change the Wampus status to dead which is implemented by wumpusisdead(1).

Variable Meaning:

X: The horizontal coordinate of wumpus, or gold, or agent,…

Y: The vertical coordinate of wumpus, or gold, or agent,…

NewX1: Room to the left.

NewX2: Room to the right

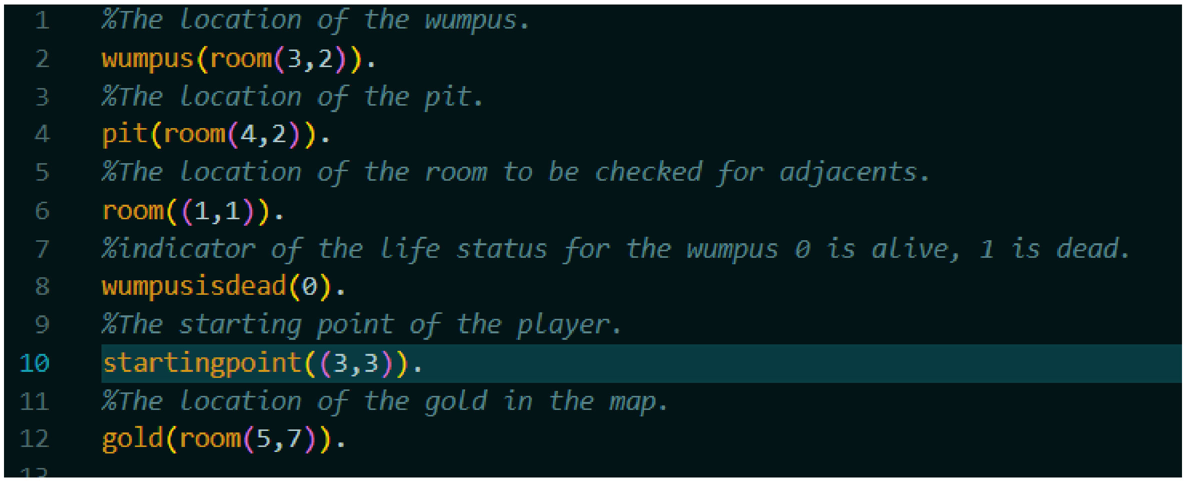
NewY1: Room downwards.

NewY2: Room upwards.

X1,Y1: Coordinates to check if there is a wumpus or gold in them.

III Experiments:

1- Let's try 8*8 Map with 1 pit:



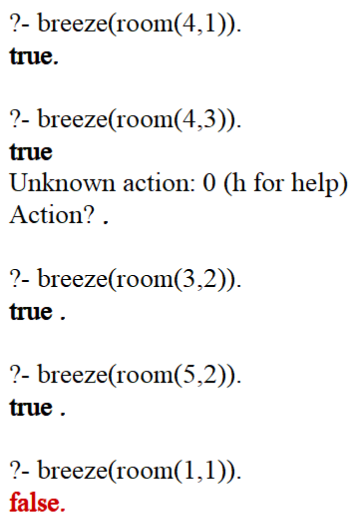
```
%The location of the wumpus.
wumpus(room(3,2)).
%The location of the pit.
pit(room(4,2)).
%The location of the room to be checked for adjacents.
room((1,1)).
%indicator of the life status for the wumpus 0 is alive, 1 is dead.
wumpusisdead(0).
%The starting point of the player.
startingpoint((3,3)).
%The location of the gold in the map.
gold(room(5,7)).
```

Let’s try some predicates:

Room function:

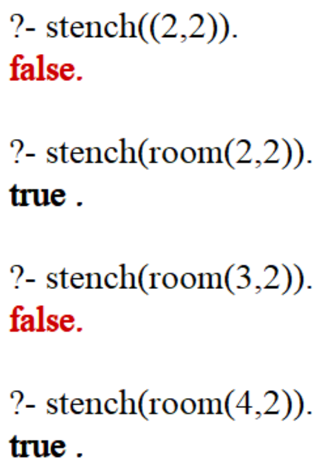
```
?- room(8,16).
false.

?- room(8,8).
true.

?- room(10,58).
false.

?- room(1,5).
true.
```

Breeze function:



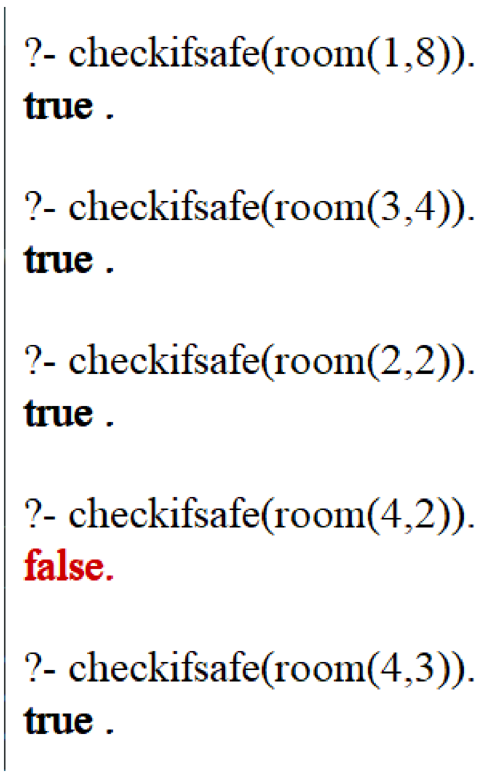
```
?- breeze(room(4,1)).
true.

?- breeze(room(4,3)).
true
Unknown action: 0 (h for help)
Action? .

?- breeze(room(3,2)).
true .

?- breeze(room(5,2)).
true .

?- breeze(room(1,1)).
false.
```

Stench function:



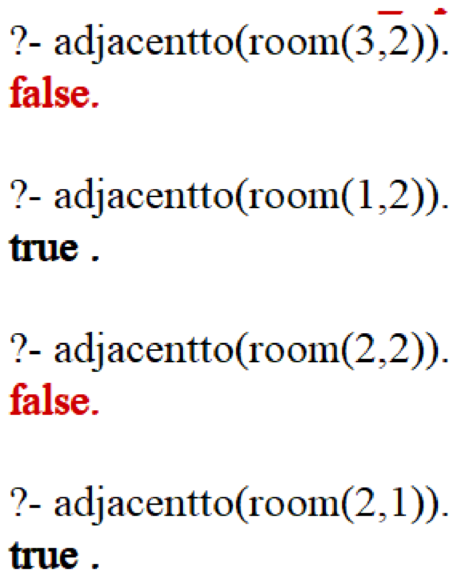
```
?- stench((2,2)).
false.

?- stench(room(2,2)).
true .

?- stench(room(3,2)).
false.

?- stench(room(4,2)).
true .
```

Safety of a room:



```
?- checkifsafe(room(1,8)).
true .

?- checkifsafe(room(3,4)).
true .

?- checkifsafe(room(2,2)).
true .

?- checkifsafe(room(4,2)).
false.

?- checkifsafe(room(4,3)).
true .
```

Adjacent rooms function (for room (1,1)):



```
?- adjacentto(room(3,2)).
false.

?- adjacentto(room(1,2)).
true .

?- adjacentto(room(2,2)).
false.

?- adjacentto(room(2,1)).
true .
```

Checking the four directions for safety:

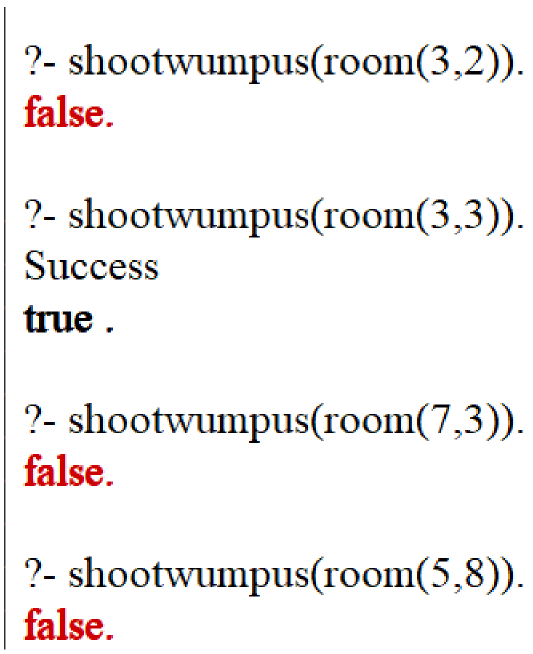
We have four directions, so we will have to check for the all of them using a starting point already declared which is startingpoint((3,3)).

```
?- checkroomonnorth().
true .

?- checkroomonsouth().
false.

?- checkroomontheleft().
true .

?- checkroomontheright().
true .
```

Test grab gold:

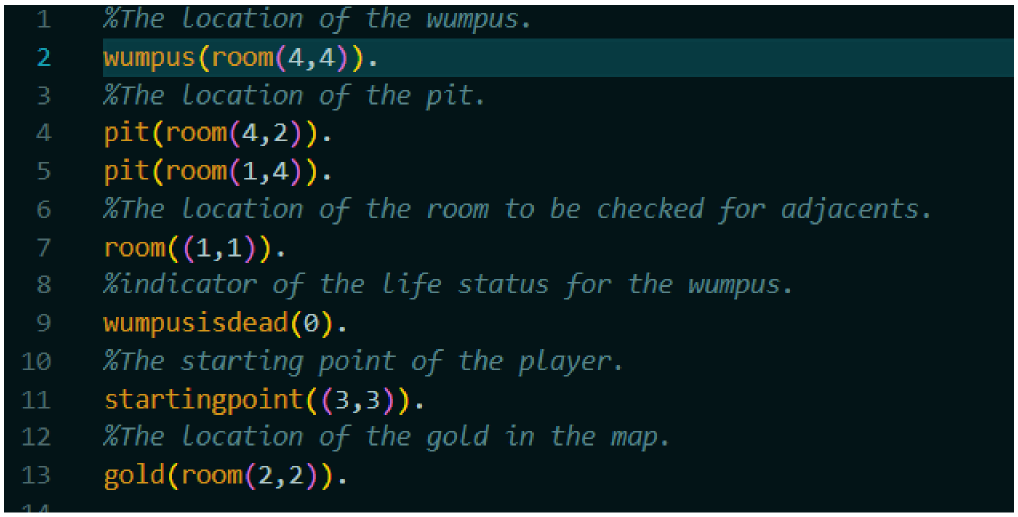
```
?- grabgold(room(1,1)).
false.

?- grabgold(room(5,1)).
false.

?- grabgold(room(5,7)).
Success
true.
```

Test Shoot Wumpus:



```
?- shootwumpus(room(3,2)).
false.

?- shootwumpus(room(3,3)).
Success
true .

?- shootwumpus(room(7,3)).
false.

?- shootwumpus(room(5,8)).
false.
```

2- Let’s try 4*4 Map with 2 pits:



```prolog
%The location of the wumpus.
wumpus(room(4,4)).
%The location of the pit.
pit(room(4,2)).
pit(room(1,4)).
%The location of the room to be checked for adjacents.
room((1,1)).
%indicator of the life status for the wumpus.
wumpusisdead(0).
%The starting point of the player.
startingpoint((3,3)).
%The location of the gold in the map.
gold(room(2,2)).
```

Room function:

```
?- room(1,1).
true.

?- room(1,5).
false.

?- room(1,2).
true.

?- room(3,2).
true.
```

Breeze function:

```
?- breeze(room(4,1)).
true.

?- breeze(room(4,2)).
false.

?- breeze(room(4,4)).
false.

?- breeze(room(2,4)).
false.

?- breeze(room(1,4)).
false.
```

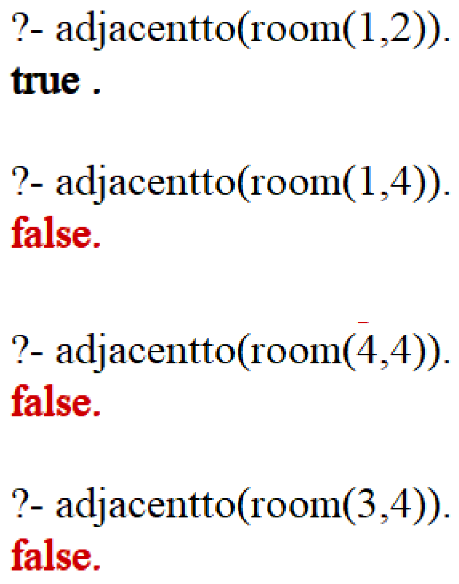
Stench function:

```
?- stench(room(1,1)).
false.

?- stench(room(4,1)).
false.

?- stench(room(4,3)).
true.

?- stench(room(3,3)).
false.
```

Safety of a room:

```
?- checkifsafe(room(3,3)).
true .

?- checkifsafe(room(1,1)).
true .

?- checkifsafe(room(2,3)).
true .

?- checkifsafe(room(4,4)).
false.
```

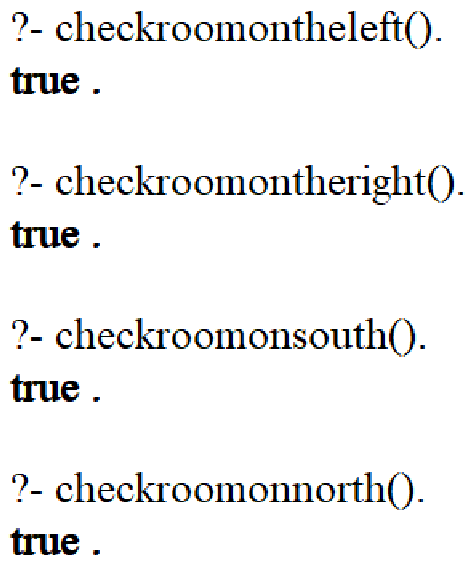
Adjacent rooms function (for room (1,1)):



```
?- adjacentto(room(1,2)).
true .

?- adjacentto(room(1,4)).
false.

?- adjacentto(room(4,4)).
false.

?- adjacentto(room(3,4)).
false.
```

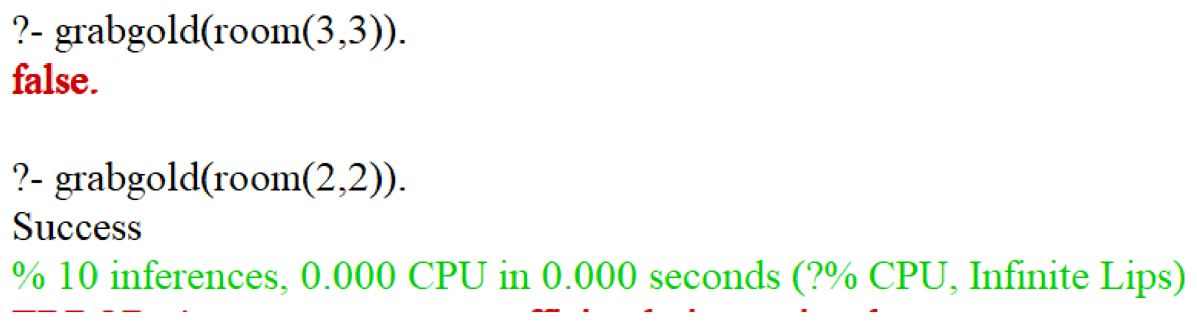
Checking the four directions for safety:

We have four directions, so we will have to check for the all of them using a starting point already declared which is startingpoint((3,3)).



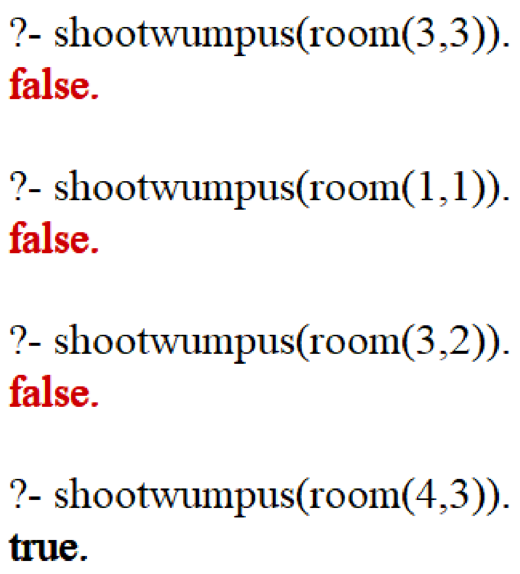
```
?- checkroomontheleft().
true .

?- checkroomontheright().
true .

?- checkroomonsouth().
true .

?- checkroomonnorth().
true .
```

Test grab gold:



```
?- grabgold(room(3,3)).
false.

?- grabgold(room(2,2)).
Success
% 10 inferences, 0.000 CPU in 0.000 seconds (?% CPU, Infinite Lips)
```

Test Shoot Wumpus:



```
?- shootwumpus(room(3,3)).
false.

?- shootwumpus(room(1,1)).
false.

?- shootwumpus(room(3,2)).
false.

?- shootwumpus(room(4,3)).
true.
```

Limitation and improvements:

This code can be greatly improved by doing things like printing the list of safe rooms, providing the beginning position, or providing a random point.

The addition of many pits can also improve this code; however, in order to do so, one must consider how to prevent problems and maintain the readability of the code.

Adding the direction in which the player is looking at can also be a nice improvement but that will change a lot on how we shoot the Wumpus since we will need to take the direction of the player into consideration.

The code needs to be improved in order to be able to handle every scenario that could occur, including two adjacent pits.

Making the code automatic can be a good method to improve quality of life as it now requires human input to determine which rooms will be visited.

I tried several times to go through the code, but there is still one issue: the Wumpus's health bar should change from 0 to 1 when the player shoots it. However, I was unable to implement this effectively and it produced too many bugs, so I eventually gave up and stuck with a value of 0.

Due to the low time efficiency of this code, I was unable to accurately determine how much time was required for each part.