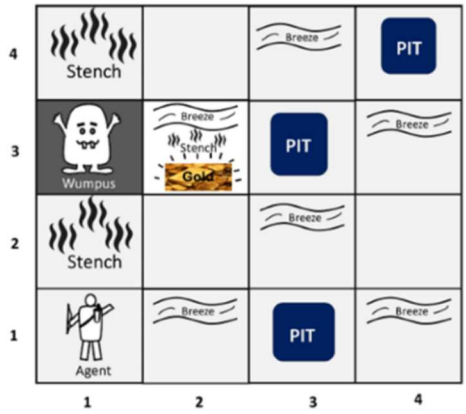
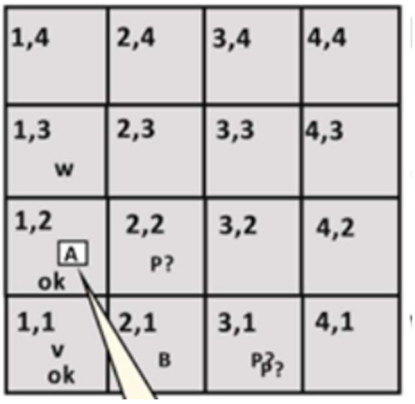
Team members: Ayoub Maimmadi, Hamza Ammad.

Report owner: Ayoub Maimmadi.

Assignment 2: The Wumpus Game.

In this project, we are asked to build a single iteration of a logical agent, that given a starting position and ASKED about one some action safety of rooms, shooting, or picking up gold will return a reply.





For this first example, we have implemented the map by assigning the rooms of this game exactly as they appear in these pictures. Meaning that the pits are in position (3,3), (4,4), and (3,1), along with the positions of the gold and the Wumpus.

```
room(4,1).
room(4,2).
room(4,3).
room(4,4).

breeze(room(2,1)).
breeze(room(4,1)).
breeze(room(3,2)).
breeze(room(2,3)).
breeze(room(4,3)).
breeze(room(3,4)).

stench(room(1,2)).
stench(room(2,3)).
stench(room(1,4)).

glitter(room(2,3)).
```

We did not explicitly code where the pit and the Wumpus will be, instead we have made a logical expression to find them of all cases in different maps, making the code upgradable.

```
wumpus(room(X,Y)) :- room(X,Y), room(A,B), stench(room(A,B)), adjacent(room(X,Y),room(A,B)).

pit(room(X,Y)) :- room(X,Y), room(A,B), breeze(room(A,B)), adjacent(room(X,Y),room(A,B)).
```

We have also included a way to automatically generate breezes and stenches given the pits and Wumpus positions in case we want to give the places of the pits and the Wumpus as rules instead of breezes and stenches.

```
breeze(room(X,Y)) :- room(X,Y), room(A,B), pit(room(A,B)), adjacent(room(X,Y),room(A,B)).

stench(room(X,Y)) :- room(X,Y), room(A,B), wumpus(room(A,B)), adjacent(room(X,Y),room(A,B)).
```

We have used our own strategy to manage adjacent rooms instead of using the ones found in GitHub, along with our own way of building the map, and of tackling tasks such as finding the best possible solutions for every position in the map.

```
adjacent(room(X,Y),room(A,B)) :- room(X,Y), room(A,B), (X=A), Z is (Y+1), (Z=B).
adjacent(room(X,Y),room(A,B)) :- room(X,Y), room(A,B), (X=A), Z is (Y-1), (Z=B).
adjacent(room(X,Y),room(A,B)) :- room(X,Y), room(A,B), (B=Y), Z is (X+1), (Z=A).
adjacent(room(X,Y),room(A,B)) :- room(X,Y), room(A,B), (B=Y), Z is (X-1), (Z=A).
```

Given the starting position of (1,1) `position(room(1,1)).`

We will try to get back all the rooms(X,Y) that are safe from the given starting position using this code.

```
safe(room(X,Y)) :- room(X,Y), candidate(room(X,Y)), room(A,B), position(room(A,B)), no_wumpus(room(X,Y)), no_pit(room(X,Y)).
```

In order for a room to be safe:

It needs to be a candidate room, and It needs to be adjacent to the starting position because we can only move once at a time:

```
candidate(room(X,Y)) :- visited(room(A,B)), position(room(A,B)), room(X,Y), adjacent(room(A,B),room(X,Y)).
```

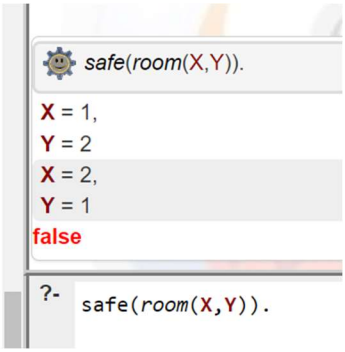
it needs to be pit and Wumpus free:

```
no_pit(room(A,B)) :- room(X,Y), adjacent(room(A,B),room(X,Y)), visited(room(X,Y)), not(breeze(room(X,Y))).
no_pit(room(A,B)) :- visited(room((A,B))).

no_wumpus(room(A,B)) :- visited(room((A,B))).
no_wumpus(room(A,B)) :- room(X,Y), adjacent(room(A,B),room(X,Y)), visited(room(X,Y)), not(stench(room(X,Y))).
```

And it needs to yield the proper results when tested:

```
34
35
36 position(room(1,1)).
37
38
39
40
41
42
43
44
```



When it comes to grabbing the gold, we need to have a glitter which mean that there is gold in that room:

```
glitter(room(2,3)).

gold(room(X,Y)) :- glitter(room(X,Y)).
```

We need our agent to be in a position to grab that gold, `position(room(2,3)).`

And then we need to ask it to grab the gold

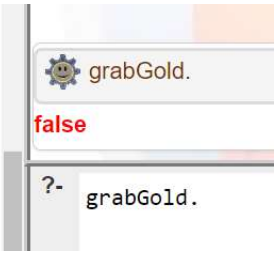
```
grabGold :- room(X,Y), gold(room(X,Y)), position(room(X,Y)).
```

and yield proper results

```
34
35
36 position(room(2,3)).
37
38
39
40
41
```

Otherwise yield false if in a position that does not contain gold

```
35
36 position(room(2,4)).
37
38
39
40
41
```



Same thing when it comes to shooting the Wumpus.

Given the agent position `position(room(1,2)).`

if he is in an adjacent room of the Wumpus's position, meaning a room with stench, the agent can shoot the Wumpus.

```
stench(room(X,Y)) :- room(X,Y), room(A,B), wumpus(room(A,B)), adjacent(room(X,Y),room(A,B)).
```

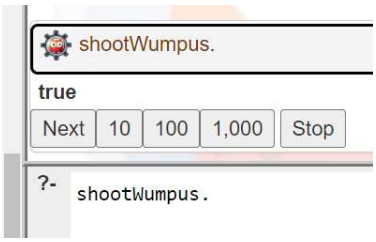
```
wumpus(room(X,Y)) :- room(X,Y), room(A,B), stench(room(A,B)), adjacent(room(X,Y),room(A,B)).
```

This automatically makes all rooms that have stenches adjacent to the room with the Wumpus.

```
shootWumpus :- room(X,Y), stench(room(X,Y)), position(room(X,Y)).
```

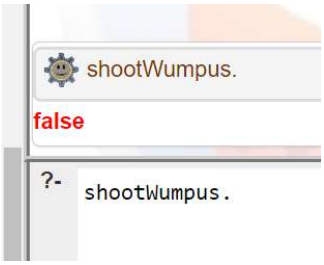
And yield proper result asked:

```
34
35
36 position(room(1,2)).
37
38
39
40
41
```



And return false if not in a position to shoot the Wumpus

```
35
36 position(room(2,2)).
37
38
39
40
41
```



Performance rate.

So far, we have gotten one correct result about the starting position, and all correct results about the secondary actions such as grab gold, shoot Wumpus, recognize adjacent rooms, recognize visited rooms, recognize where the gold would be (when detecting glitter), recognizing what does the stench mean, and what does the breeze mean, recognize candidate rooms to move into depending on the position.

So, secondary actions such as grab gold and shoot Wumpus are potentially 100% accurate.

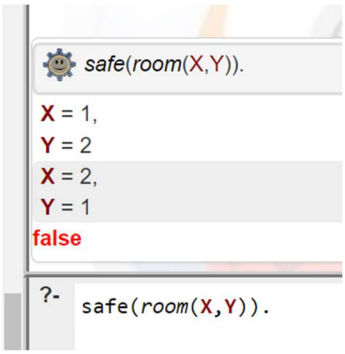
The movement and recognition (adjacent, and Wumpus/pit existence) actions yielded correct results by my tests 100% of the time. However, I really doubt that personally in the grand scheme of things, that's why I will be giving it only 95% accuracy. I may not have accounted for something.

Now let's computer the performance rate when it comes to finding save rooms.

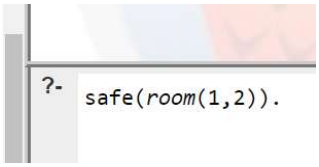
Starting different positions, let's try to get the safe rooms

Test1: initial starting position (1.1)

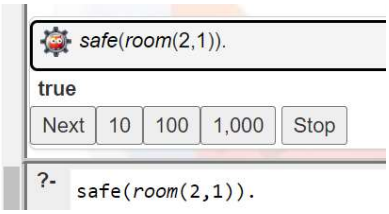
```
position(room(1,1)).
```




```
position(room(1,1)).
```



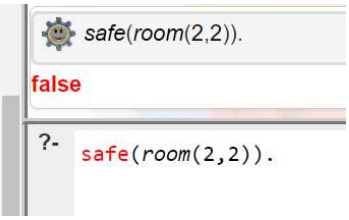

```
position(room(1,1)).
```



Position (1,2)

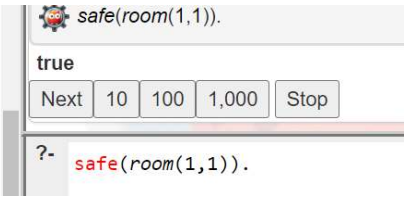
```
position(room(1,2)).
```


```
position(room(1,2)).
```

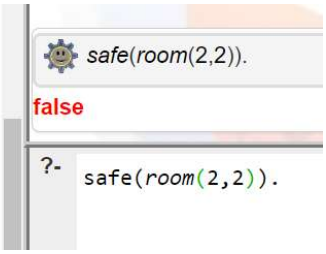


Position (2,1)

```
33
34
35
36
37 position(room(2,1)).
38
39
```




```
33
34
35
36
37 position(room(2,1)).
38
39
40
```



……..

Shortcomings: For this particular code, the agent recognizes that the room (2,2) is not safe, however, it doesn't take into consideration that we have previously visited (1,2) and found it not safe because there was a stench, and we don't find a stench in (2,1) meaning there couldn't be a Wumpus there nor a pit because there was no breeze in (1,2)

When I ran all the tests for all configurations, I got a result of 13/16 which means that my agent is 81.25% accurate when it comes to detecting safe and non-safe rooms for average maps with 4x4 positions.

95% accurate when it comes to map recognition and awareness (detecting candidate rooms, adjacent rooms, no Wumpus rooms, no pit rooms) when asked directly given a position.

100% accurate when it comes to shooting the Wumpus, grabbing gold, and other secondary tasks.

Conclusion

Our agent is the not most intelligent one given the stats it resulted. However, if we put it in a position, and give it the necessary information which he should have gathered himself, the agent will give the proper answer to every question asked. But, since we are not asked to code a full on logical agent by himself but one that given a position should yield good answers, I would say that this agent gets the job done properly as given and asked.