Project2: Wampus world

Oumaima laghzaoui **Teammate:Latija jbarek**

Introduction:

The Wumpus world is a simple world example to illustrate the worth of a knowledge -based agent and to represent knowledge.

In this project we worked on coding a logical agent for the Wumpus game.

The agent aims to kill the Wumpus, during this process the agent tries also to grab the gold to increase his score. The game ends if either the agent dies or the agent shoots the Wumpus.

Conditions:

The rooms adjacent to the Wumpus are smelly so if the agent enters the room of stench, he can recognize that the Wampus exists in the adjacent rooms.

The rooms adjacent to pits have breeze so whenever the agent enters breeze rooms, he will recognize that the adjacent rooms contain a pit.

There will be glitter in the room if only if the room has gold.

Predicates:

a. Room(X,Y) the room in position (x,y). You can also use a list [x,y]
b. Breeze(R(X,Y)) there is a breeze in room R(X,Y). Can also be Breeze([x,y])
c. Pit((R(X,Y)) there is a pit in room R(X,Y)
d. Wumpus(R(X,Y) the Wumpus in room R(X,Y)

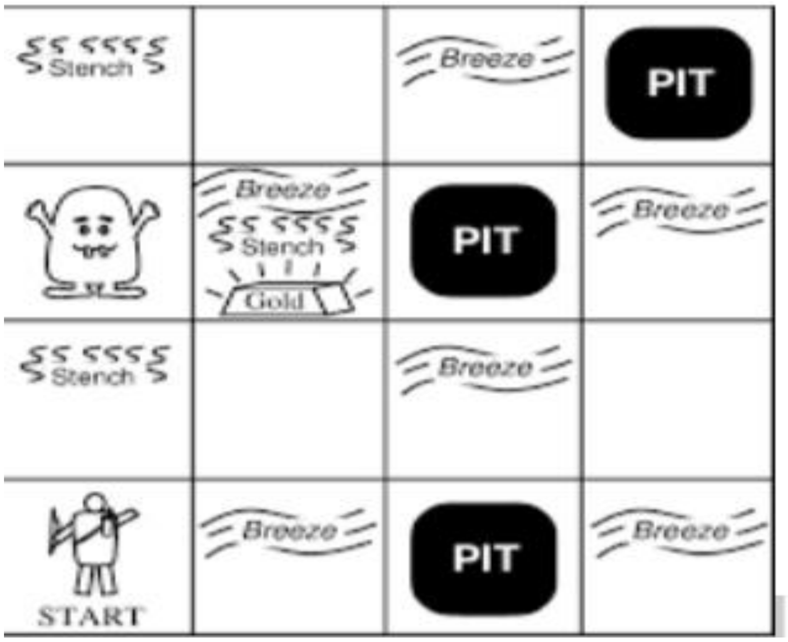
e.Stench(R(X,Y))roomR(X,Y)stenches

f.Gold(R(X, Y)) there is gold in roomR(X,Y)

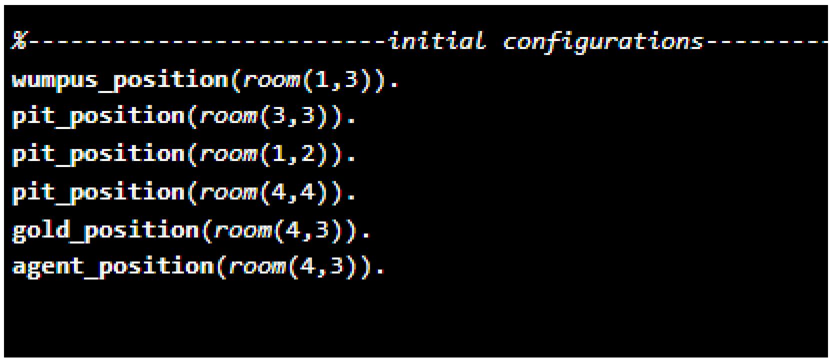
g.AdjacentTo(R(X, Y),R(ZT)) room(T,Z) is adjacent to room R(X, Y)
h.Safe(R(X,Y))room R(X?Y) is safe

i.GrabGold()grab the gold in the current room

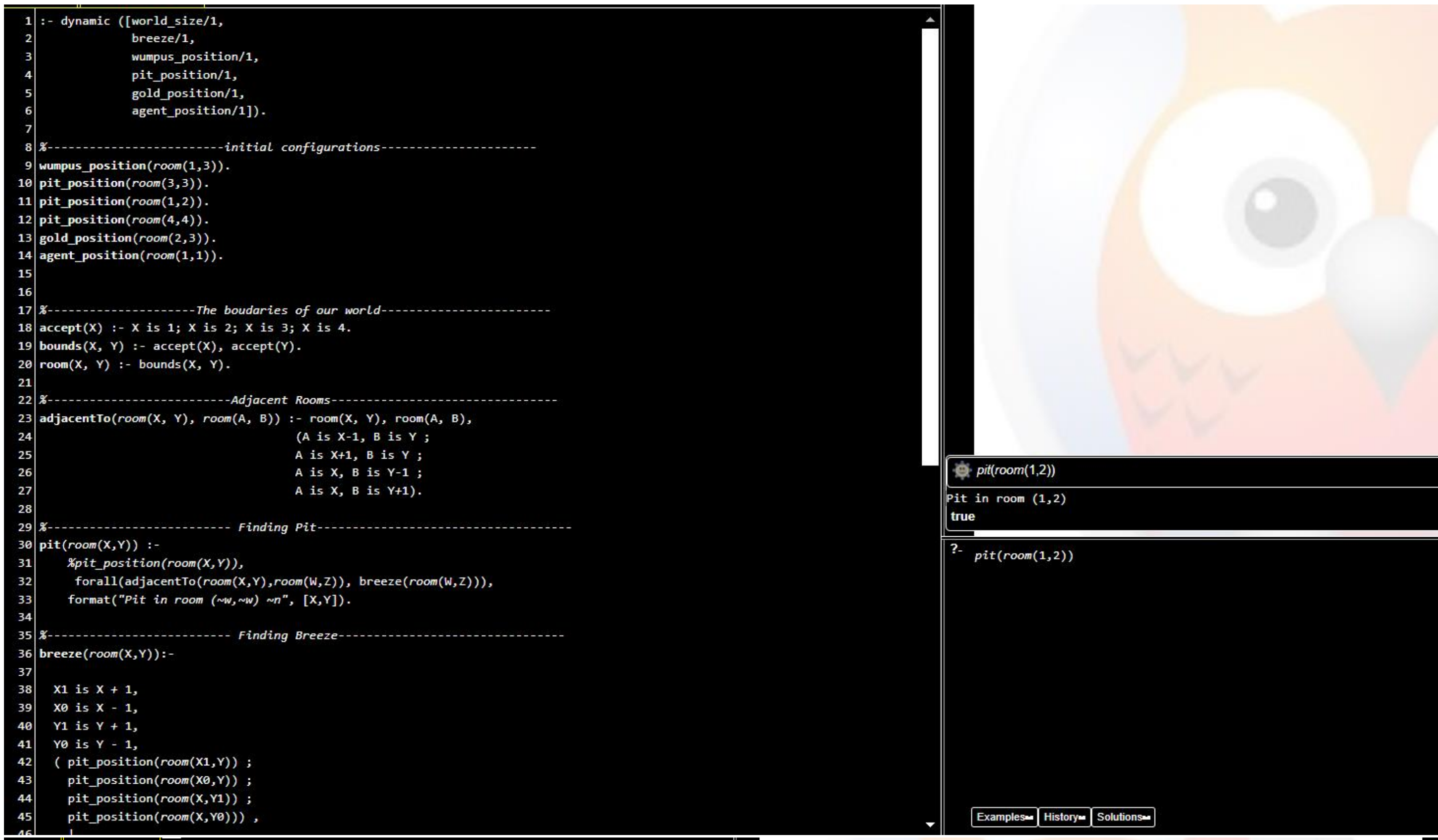
j.shootWumpus(R(X, Y)) shoot the wumpus in room R(X, Y) only from adjacent rooms



Let's try the initial configurations:

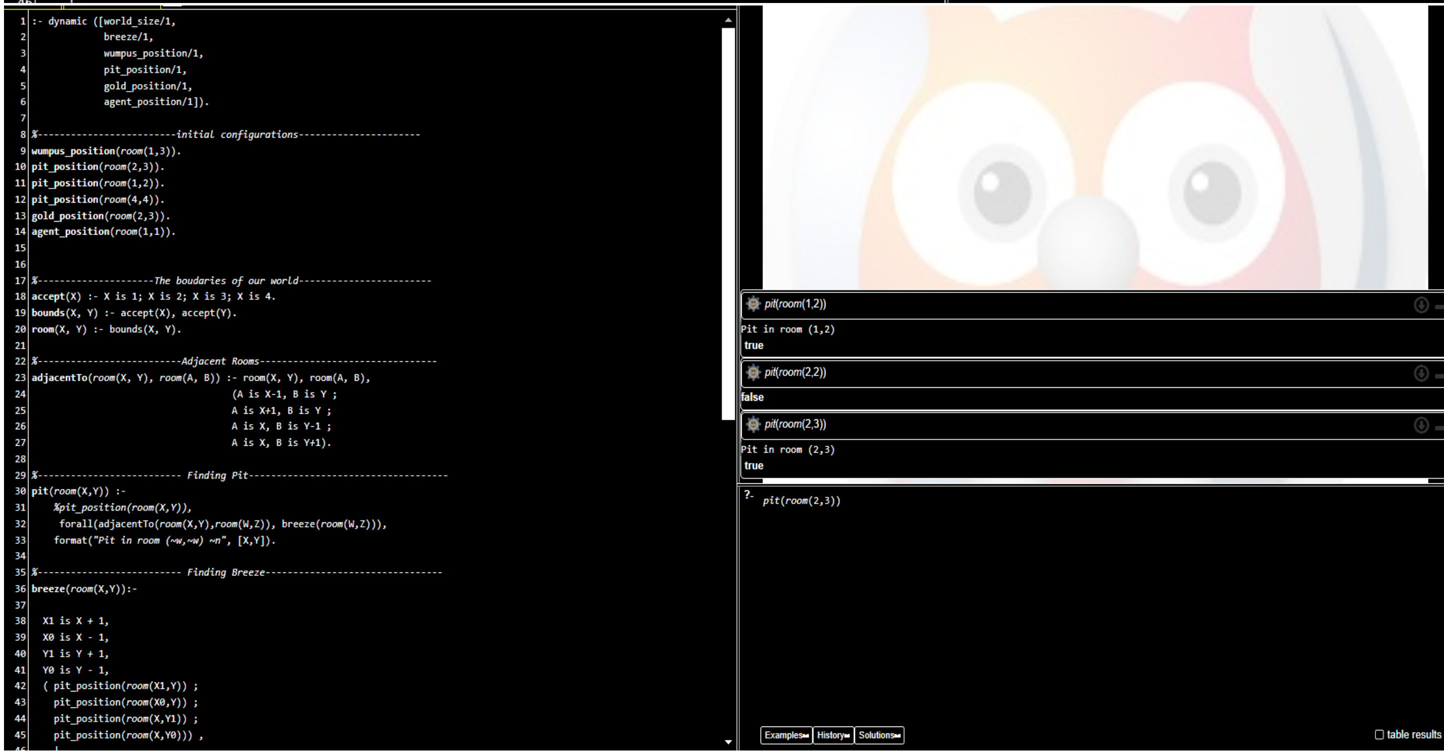


```
%------------------------initial configurations--------
wumpus_position(room(1,3)).
pit_position(room(3,3)).
pit_position(room(1,2)).
pit_position(room(4,4)).
gold_position(room(4,3)).
agent_position(room(4,3)).
```



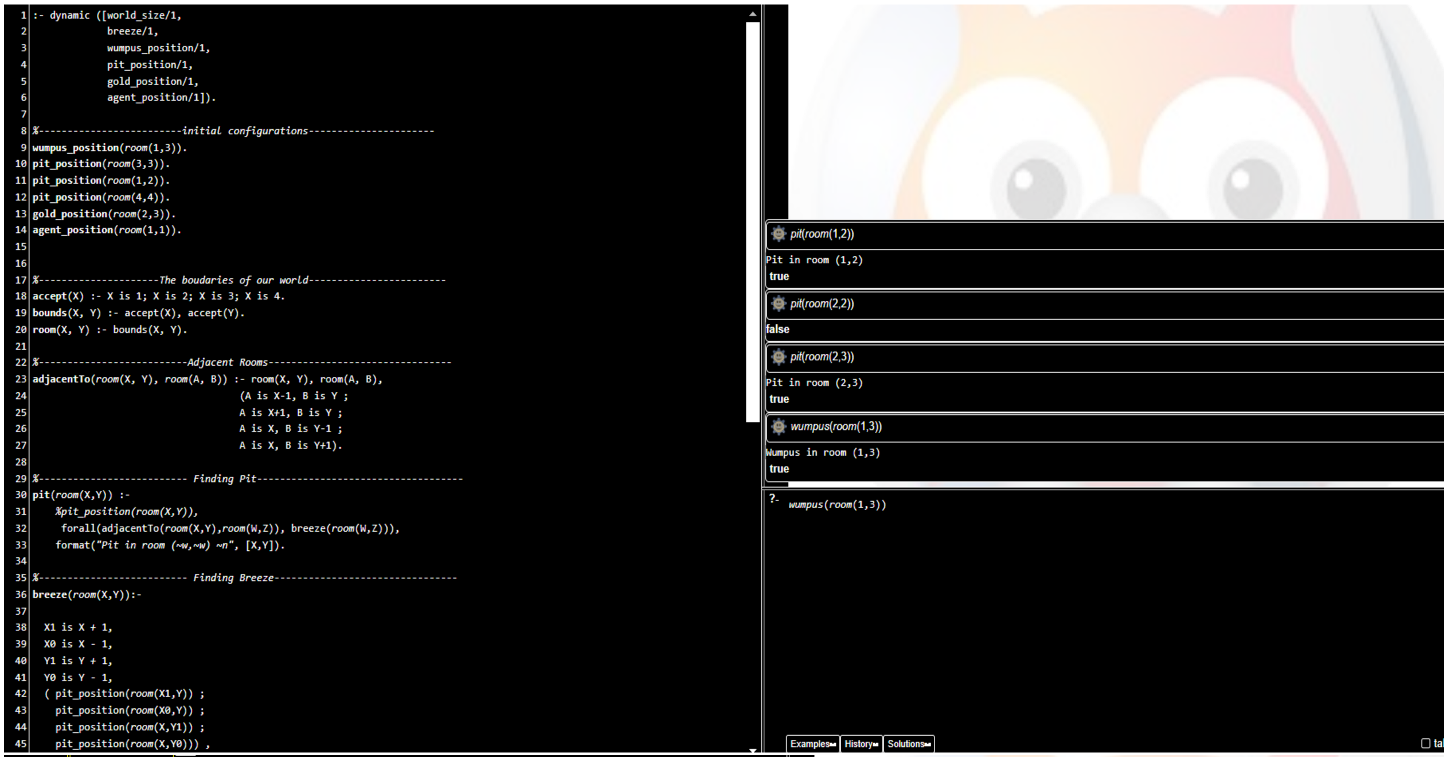
```
:- dynamic ([world_size/1,
             breeze/1,
             wumpus_position/1,
             pit_position/1,
             gold_position/1,
             agent_position/1]).

%------------------------initial configurations---------------------
wumpus_position(room(1,3)).
pit_position(room(3,3)).
pit_position(room(1,2)).
pit_position(room(4,4)).
gold_position(room(2,3)).
agent_position(room(1,1)).


%--------------------The boudaries of our world-----------------------
accept(X) :- X is 1; X is 2; X is 3; X is 4.
bounds(X, Y) :- accept(X), accept(Y).
room(X, Y) :- bounds(X, Y).

%-------------------------Adjacent Rooms-------------------------------
adjacentTo(room(X, Y), room(A, B)) :- room(X, Y), room(A, B),
                                    (A is X-1, B is Y ;
                                    A is X+1, B is Y ;
                                    A is X, B is Y-1 ;
                                    A is X, B is Y+1).

%------------------------- Finding Pit-----------------------------------
pit(room(X,Y)) :-
    %pit_position(room(X,Y)),
     forall(adjacentTo(room(X,Y),room(W,Z)), breeze(room(W,Z))),
    format("Pit in room (~w,~w) ~n", [X,Y]).

%------------------------- Finding Breeze-------------------------------
breeze(room(X,Y)):-

  X1 is X + 1,
  X0 is X - 1,
  Y1 is Y + 1,
  Y0 is Y - 1,
  ( pit_position(room(X1,Y)) ;
    pit_position(room(X0,Y)) ;
    pit_position(room(X,Y1)) ;
    pit_position(room(X,Y0))) ,
```

```
pit(room(1,2))
Pit in room (1,2)
true

?- pit(room(1,2))
```

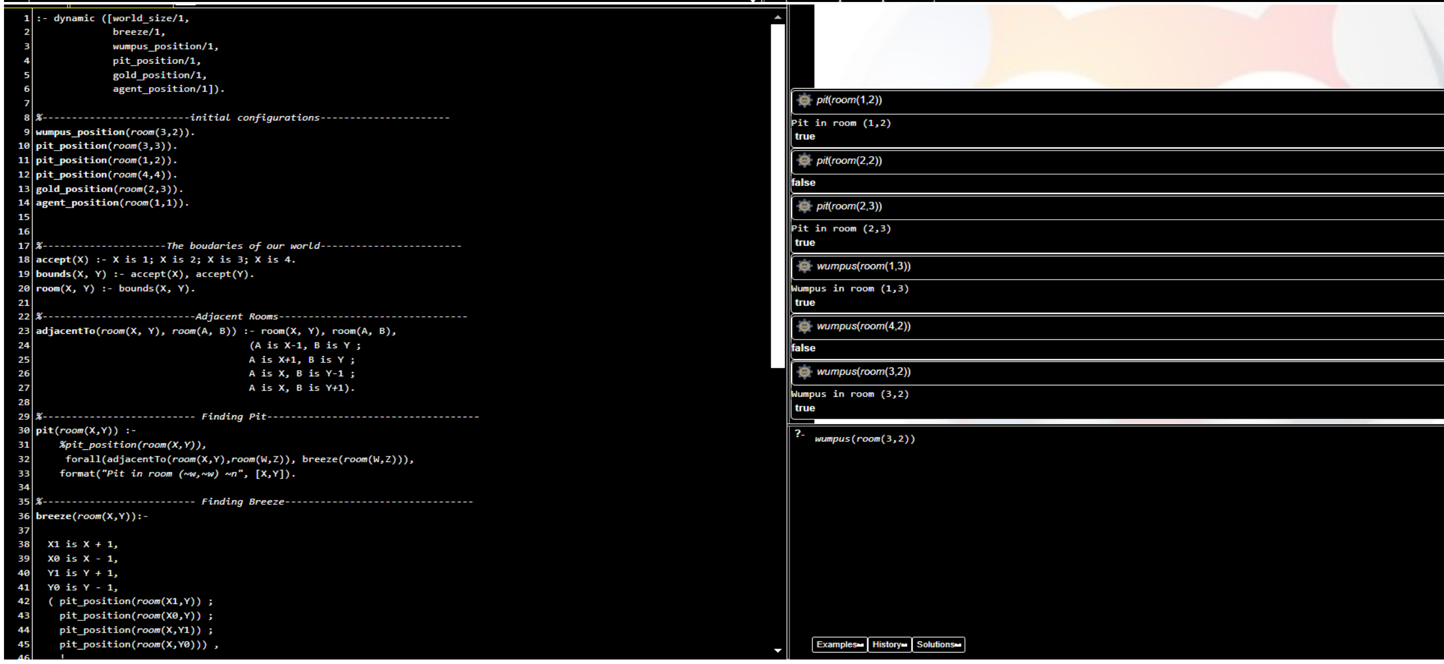


```
:- dynamic ([world_size/1,
             breeze/1,
             wumpus_position/1,
             pit_position/1,
             gold_position/1,
             agent_position/1]).

%------------------------initial configurations---------------------
wumpus_position(room(1,3)).
pit_position(room(2,3)).
pit_position(room(1,2)).
pit_position(room(4,4)).
gold_position(room(2,3)).
agent_position(room(1,1)).


%--------------------The boudaries of our world-----------------------
accept(X) :- X is 1; X is 2; X is 3; X is 4.
bounds(X, Y) :- accept(X), accept(Y).
room(X, Y) :- bounds(X, Y).

%-------------------------Adjacent Rooms-------------------------------
adjacentTo(room(X, Y), room(A, B)) :- room(X, Y), room(A, B),
                                    (A is X-1, B is Y ;
                                    A is X+1, B is Y ;
                                    A is X, B is Y-1 ;
                                    A is X, B is Y+1).

%------------------------- Finding Pit-----------------------------------
pit(room(X,Y)) :-
    %pit_position(room(X,Y)),
     forall(adjacentTo(room(X,Y),room(W,Z)), breeze(room(W,Z))),
    format("Pit in room (~w,~w) ~n", [X,Y]).

%------------------------- Finding Breeze-------------------------------
breeze(room(X,Y)):-

  X1 is X + 1,
  X0 is X - 1,
  Y1 is Y + 1,
  Y0 is Y - 1,
  ( pit_position(room(X1,Y)) ;
    pit_position(room(X0,Y)) ;
    pit_position(room(X,Y1)) ;
    pit_position(room(X,Y0))) ,
```

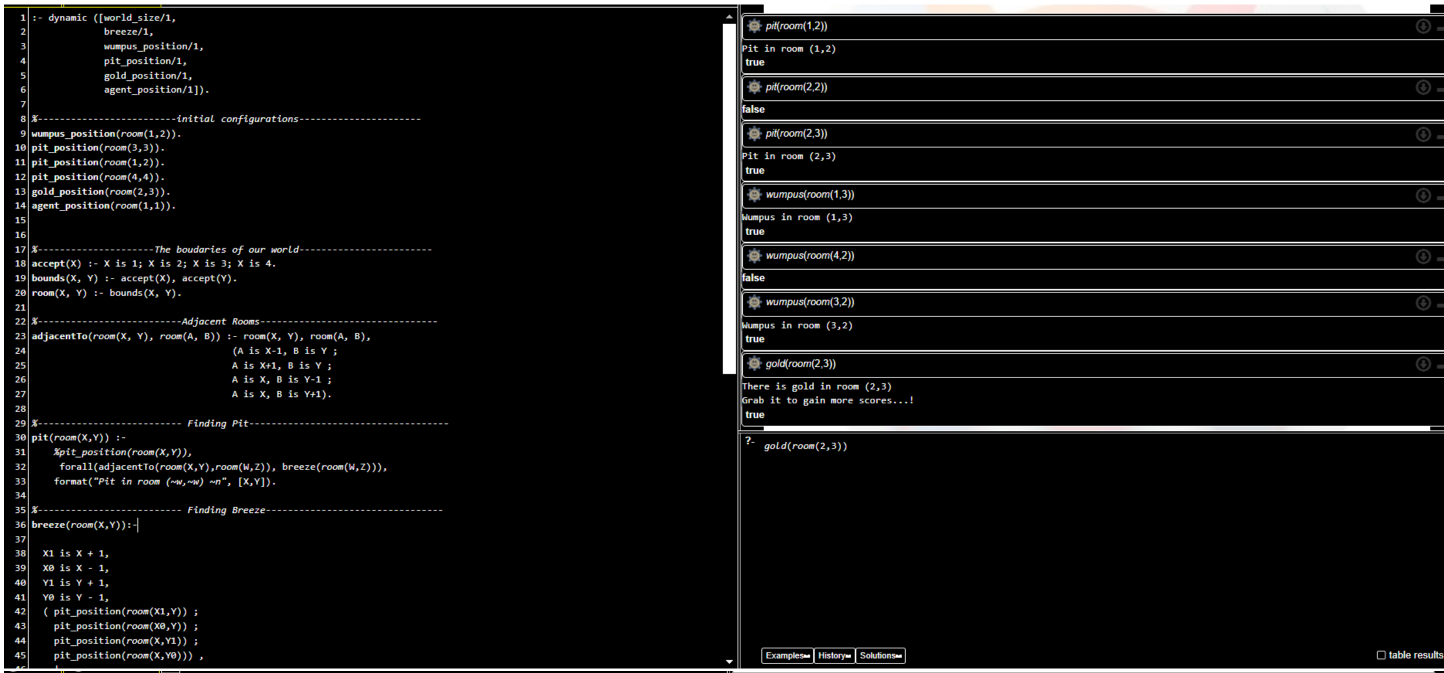
```
pit(room(1,2))
Pit in room (1,2)
true

pit(room(2,2))
false

pit(room(2,3))
Pit in room (2,3)
true

?- pit(room(2,3))
```


```prolog
:- dynamic ([world_size/1,
            breeze/1,
            wumpus_position/1,
            pit_position/1,
            gold_position/1,
            agent_position/1]).

%------------------------initial configurations---------------------
wumpus_position(room(1,3)).
pit_position(room(3,3)).
pit_position(room(1,2)).
pit_position(room(4,4)).
gold_position(room(2,3)).
agent_position(room(1,1)).


%--------------------The boudaries of our world-----------------------
accept(X) :- X is 1; X is 2; X is 3; X is 4.
bounds(X, Y) :- accept(X), accept(Y).
room(X, Y) :- bounds(X, Y).

%------------------------Adjacent Rooms-----------------------------
adjacentTo(room(X, Y), room(A, B)) :- room(X, Y), room(A, B),
                                    (A is X-1, B is Y ;
                                    A is X+1, B is Y ;
                                    A is X, B is Y-1 ;
                                    A is X, B is Y+1).

%------------------------ Finding Pit----------------------------------
pit(room(X,Y)) :-
    %pit_position(room(X,Y)),
     forall(adjacentTo(room(X,Y),room(W,Z)), breeze(room(W,Z))),
    format("Pit in room (~w,~w) ~n", [X,Y]).

%------------------------ Finding Breeze------------------------------
breeze(room(X,Y)):-

  X1 is X + 1,
  X0 is X - 1,
  Y1 is Y + 1,
  Y0 is Y - 1,
  ( pit_position(room(X1,Y)) ;
    pit_position(room(X0,Y)) ;
    pit_position(room(X,Y1)) ;
    pit_position(room(X,Y0))) ,
```

```
pit(room(1,2))
Pit in room (1,2)
true
pit(room(2,2))
false
pit(room(2,3))
Pit in room (2,3)
true
wumpus(room(1,3))
Wumpus in room (1,3)
true

?- wumpus(room(1,3))
```


```prolog
:- dynamic ([world_size/1,
            breeze/1,
            wumpus_position/1,
            pit_position/1,
            gold_position/1,
            agent_position/1]).

%------------------------initial configurations---------------------
wumpus_position(room(3,2)).
pit_position(room(3,3)).
pit_position(room(1,2)).
pit_position(room(4,4)).
gold_position(room(2,3)).
agent_position(room(1,1)).


%--------------------The boudaries of our world-----------------------
accept(X) :- X is 1; X is 2; X is 3; X is 4.
bounds(X, Y) :- accept(X), accept(Y).
room(X, Y) :- bounds(X, Y).

%------------------------Adjacent Rooms-----------------------------
adjacentTo(room(X, Y), room(A, B)) :- room(X, Y), room(A, B),
                                    (A is X-1, B is Y ;
                                    A is X+1, B is Y ;
                                    A is X, B is Y-1 ;
                                    A is X, B is Y+1).

%------------------------ Finding Pit----------------------------------
pit(room(X,Y)) :-
    %pit_position(room(X,Y)),
     forall(adjacentTo(room(X,Y),room(W,Z)), breeze(room(W,Z))),
    format("Pit in room (~w,~w) ~n", [X,Y]).

%------------------------ Finding Breeze------------------------------
breeze(room(X,Y)):-

  X1 is X + 1,
  X0 is X - 1,
  Y1 is Y + 1,
  Y0 is Y - 1,
  ( pit_position(room(X1,Y)) ;
    pit_position(room(X0,Y)) ;
    pit_position(room(X,Y1)) ;
    pit_position(room(X,Y0))) ,
```

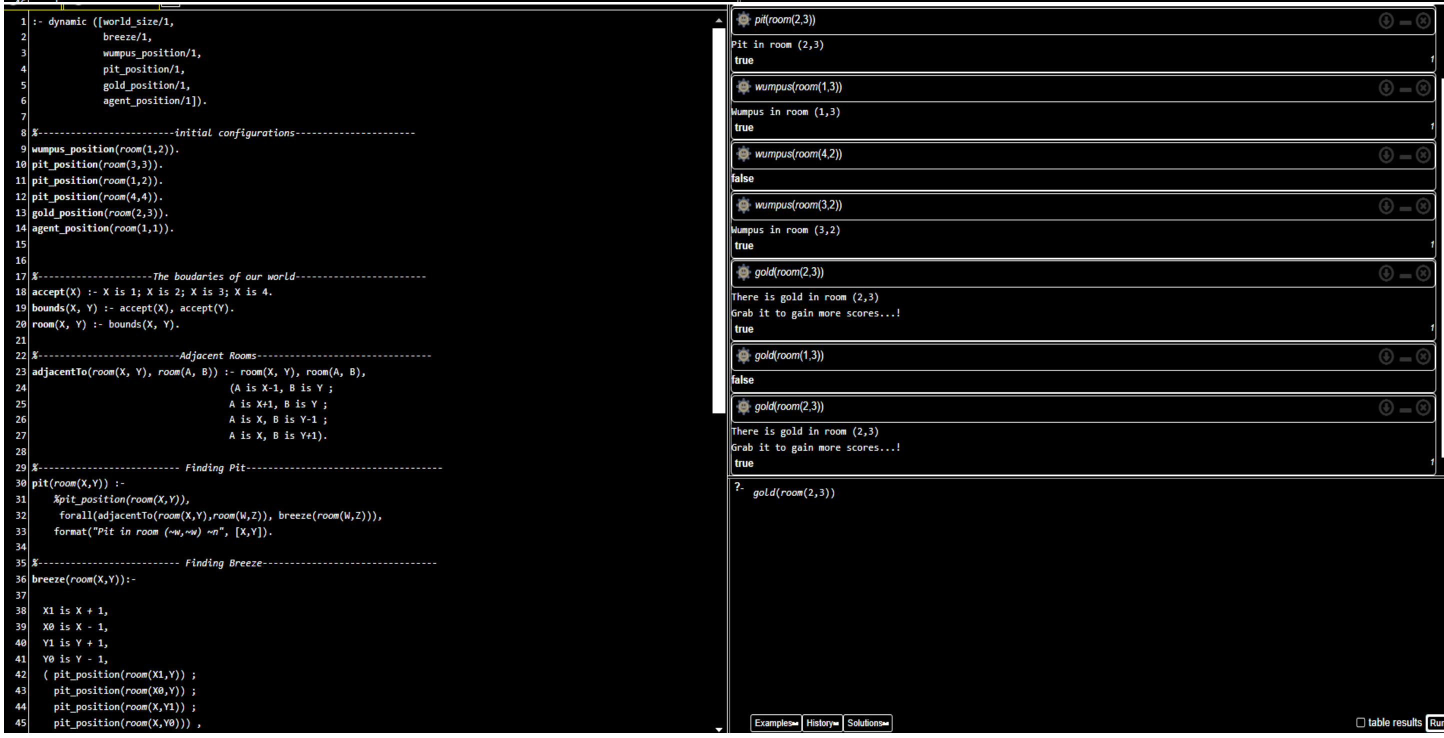
```
pit(room(1,2))
Pit in room (1,2)
true
pit(room(2,2))
false
pit(room(2,3))
Pit in room (2,3)
true
wumpus(room(1,3))
Wumpus in room (1,3)
true
wumpus(room(4,2))
false
wumpus(room(3,2))
Wumpus in room (3,2)
true

?- wumpus(room(3,2))
```



```
:- dynamic ([world_size/1,
            breeze/1,
            wumpus_position/1,
            pit_position/1,
            gold_position/1,
            agent_position/1]).

%------------------------initial configurations---------------------
wumpus_position(room(1,2)).
pit_position(room(3,3)).
pit_position(room(1,2)).
pit_position(room(4,4)).
gold_position(room(2,3)).
agent_position(room(1,1)).


%--------------------The boudaries of our world----------------------
accept(X) :- X is 1; X is 2; X is 3; X is 4.
bounds(X, Y) :- accept(X), accept(Y).
room(X, Y) :- bounds(X, Y).

%------------------------Adjacent Rooms------------------------------
adjacentTo(room(X, Y), room(A, B)) :- room(X, Y), room(A, B),
                                     (A is X-1, B is Y ;
                                     A is X+1, B is Y ;
                                     A is X, B is Y-1 ;
                                     A is X, B is Y+1).

%------------------------ Finding Pit--------------------------------
pit(room(X,Y)) :-
    %pit_position(room(X,Y)),
     forall(adjacentTo(room(X,Y),room(W,Z)), breeze(room(W,Z))),
    format("Pit in room (~w,~w) ~n", [X,Y]).

%------------------------ Finding Breeze-----------------------------
breeze(room(X,Y)):-

  X1 is X + 1,
  X0 is X - 1,
  Y1 is Y + 1,
  Y0 is Y - 1,
  ( pit_position(room(X1,Y)) ;
    pit_position(room(X0,Y)) ;
    pit_position(room(X,Y1)) ;
    pit_position(room(X,Y0))) ,
```

```
pit(room(1,2))
Pit in room (1,2)
true

pit(room(2,2))
false

pit(room(2,3))
Pit in room (2,3)
true

wumpus(room(1,3))
Wumpus in room (1,3)
true

wumpus(room(4,2))
false

wumpus(room(3,2))
Wumpus in room (3,2)
true

gold(room(2,3))
There is gold in room (2,3)
Grab it to gain more scores...!
true

?- gold(room(2,3))
```

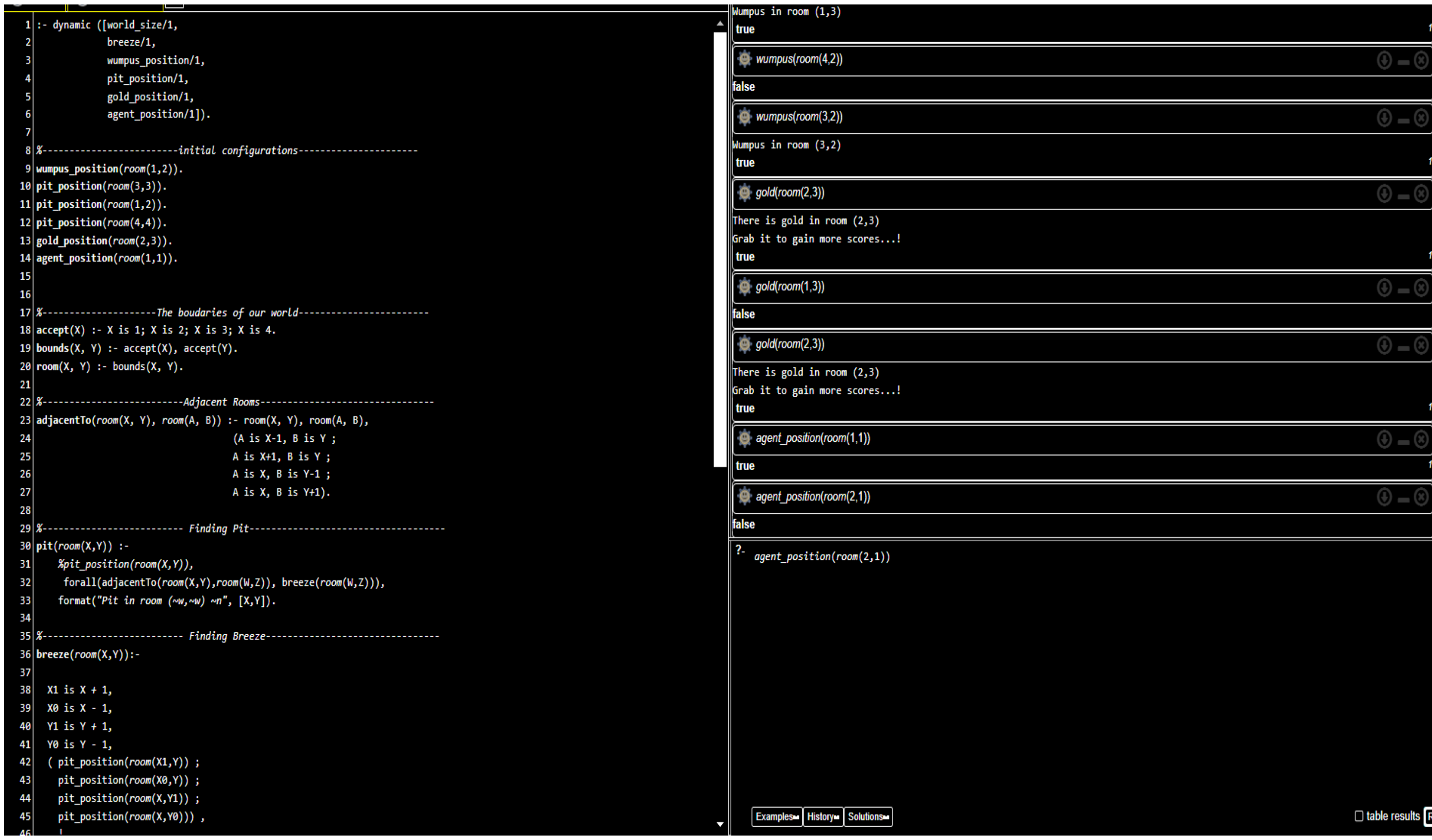
Examples History Solutions table results



```
:- dynamic ([world_size/1,
            breeze/1,
            wumpus_position/1,
            pit_position/1,
            gold_position/1,
            agent_position/1]).

%------------------------initial configurations---------------------
wumpus_position(room(1,2)).
pit_position(room(3,3)).
pit_position(room(1,2)).
pit_position(room(4,4)).
gold_position(room(2,3)).
agent_position(room(1,1)).


%--------------------The boudaries of our world----------------------
accept(X) :- X is 1; X is 2; X is 3; X is 4.
bounds(X, Y) :- accept(X), accept(Y).
room(X, Y) :- bounds(X, Y).

%------------------------Adjacent Rooms------------------------------
adjacentTo(room(X, Y), room(A, B)) :- room(X, Y), room(A, B),
                                     (A is X-1, B is Y ;
                                     A is X+1, B is Y ;
                                     A is X, B is Y-1 ;
                                     A is X, B is Y+1).

%------------------------ Finding Pit--------------------------------
pit(room(X,Y)) :-
    %pit_position(room(X,Y)),
     forall(adjacentTo(room(X,Y),room(W,Z)), breeze(room(W,Z))),
    format("Pit in room (~w,~w) ~n", [X,Y]).

%------------------------ Finding Breeze-----------------------------
breeze(room(X,Y)):-

  X1 is X + 1,
  X0 is X - 1,
  Y1 is Y + 1,
  Y0 is Y - 1,
  ( pit_position(room(X1,Y)) ;
    pit_position(room(X0,Y)) ;
    pit_position(room(X,Y1)) ;
    pit_position(room(X,Y0))) ,
```

```
pit(room(2,3))
Pit in room (2,3)
true

wumpus(room(1,3))
Wumpus in room (1,3)
true

wumpus(room(4,2))
false

wumpus(room(3,2))
Wumpus in room (3,2)
true

gold(room(2,3))
There is gold in room (2,3)
Grab it to gain more scores...!
true

gold(room(1,3))
false

gold(room(2,3))
There is gold in room (2,3)
Grab it to gain more scores...!
true

?- gold(room(2,3))
```

Examples History Solutions table results

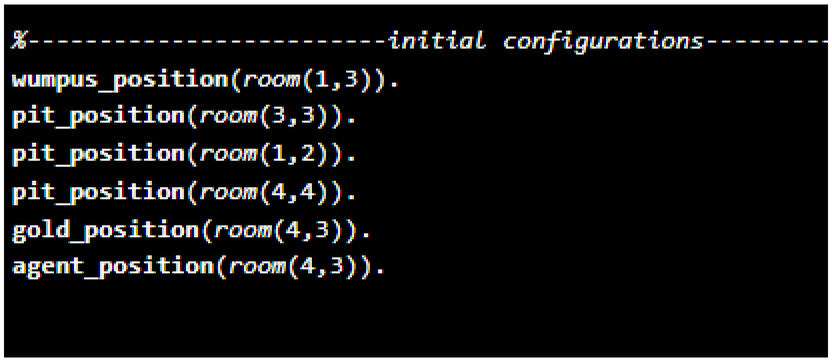


```
:- dynamic ([world_size/1,
            breeze/1,
            wumpus_position/1,
            pit_position/1,
            gold_position/1,
            agent_position/1]).

%-----------------------initial configurations---------------------
wumpus_position(room(1,2)).
pit_position(room(3,3)).
pit_position(room(1,2)).
pit_position(room(4,4)).
gold_position(room(2,3)).
agent_position(room(1,1)).


%--------------------The boudaries of our world-----------------------
accept(X) :- X is 1; X is 2; X is 3; X is 4.
bounds(X, Y) :- accept(X), accept(Y).
room(X, Y) :- bounds(X, Y).

%------------------------Adjacent Rooms-------------------------------
adjacentTo(room(X, Y), room(A, B)) :- room(X, Y), room(A, B),
                                  (A is X-1, B is Y ;
                                  A is X+1, B is Y ;
                                  A is X, B is Y-1 ;
                                  A is X, B is Y+1).

%------------------------ Finding Pit----------------------------------
pit(room(X,Y)) :-
   %pit_position(room(X,Y)),
    forall(adjacentTo(room(X,Y),room(W,Z)), breeze(room(W,Z))),
   format("Pit in room (~w,~w) ~n", [X,Y]).

%------------------------ Finding Breeze------------------------------
breeze(room(X,Y)):-

  X1 is X + 1,
  X0 is X - 1,
  Y1 is Y + 1,
  Y0 is Y - 1,
  ( pit_position(room(X1,Y)) ;
    pit_position(room(X0,Y)) ;
    pit_position(room(X,Y1)) ;
    pit_position(room(X,Y0))) ,
    !
```

```
Wumpus in room (1,3)
true
wumpus(room(4,2))
false
wumpus(room(3,2))
Wumpus in room (3,2)
true
gold(room(2,3))
There is gold in room (2,3)
Grab it to gain more scores...!
true
gold(room(1,3))
false
gold(room(2,3))
There is gold in room (2,3)
Grab it to gain more scores...!
true
agent_position(room(1,1))
true
agent_position(room(2,1))
false
?- agent_position(room(2,1))
```

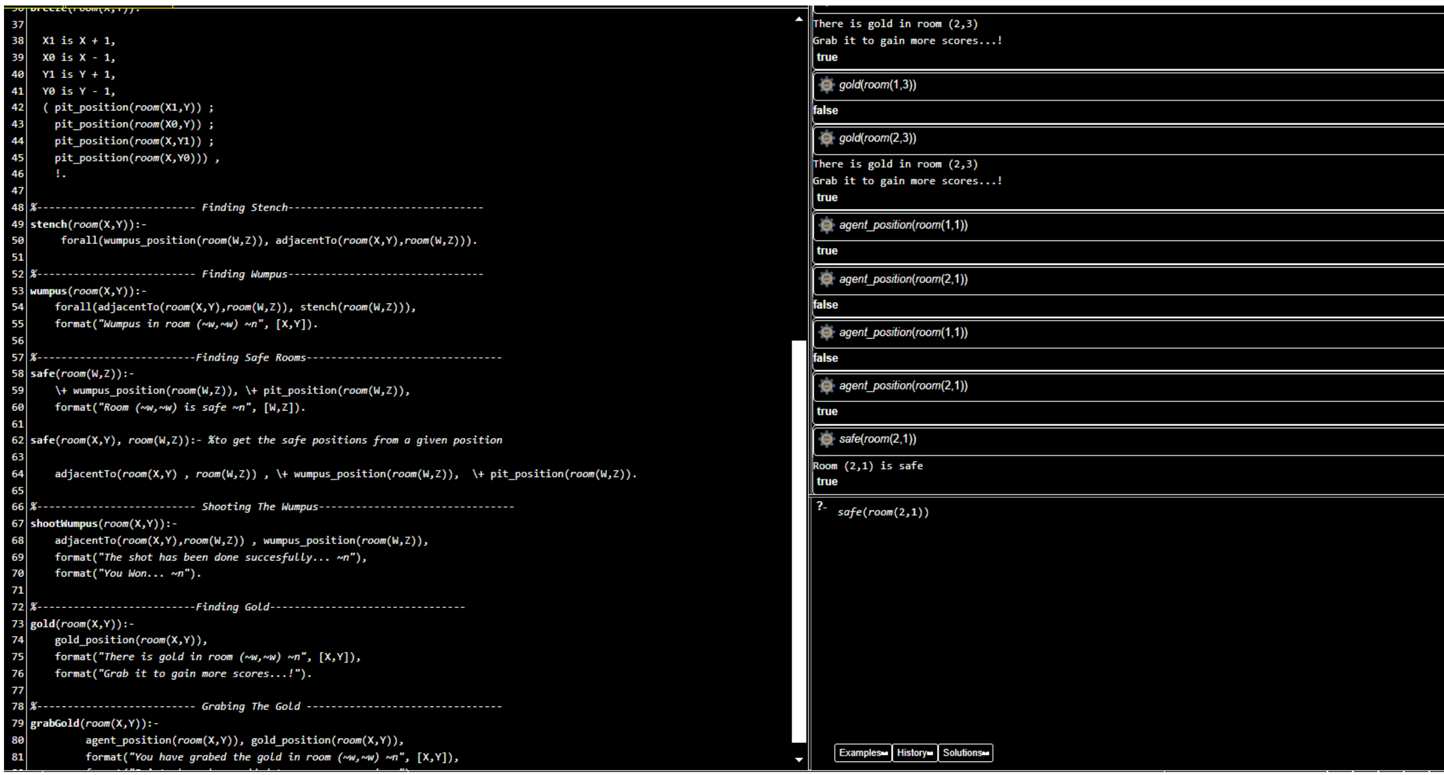
Now, we will try the functions with the initial configurations and then modify the configurations and try the configurations again:

Initial configurations:



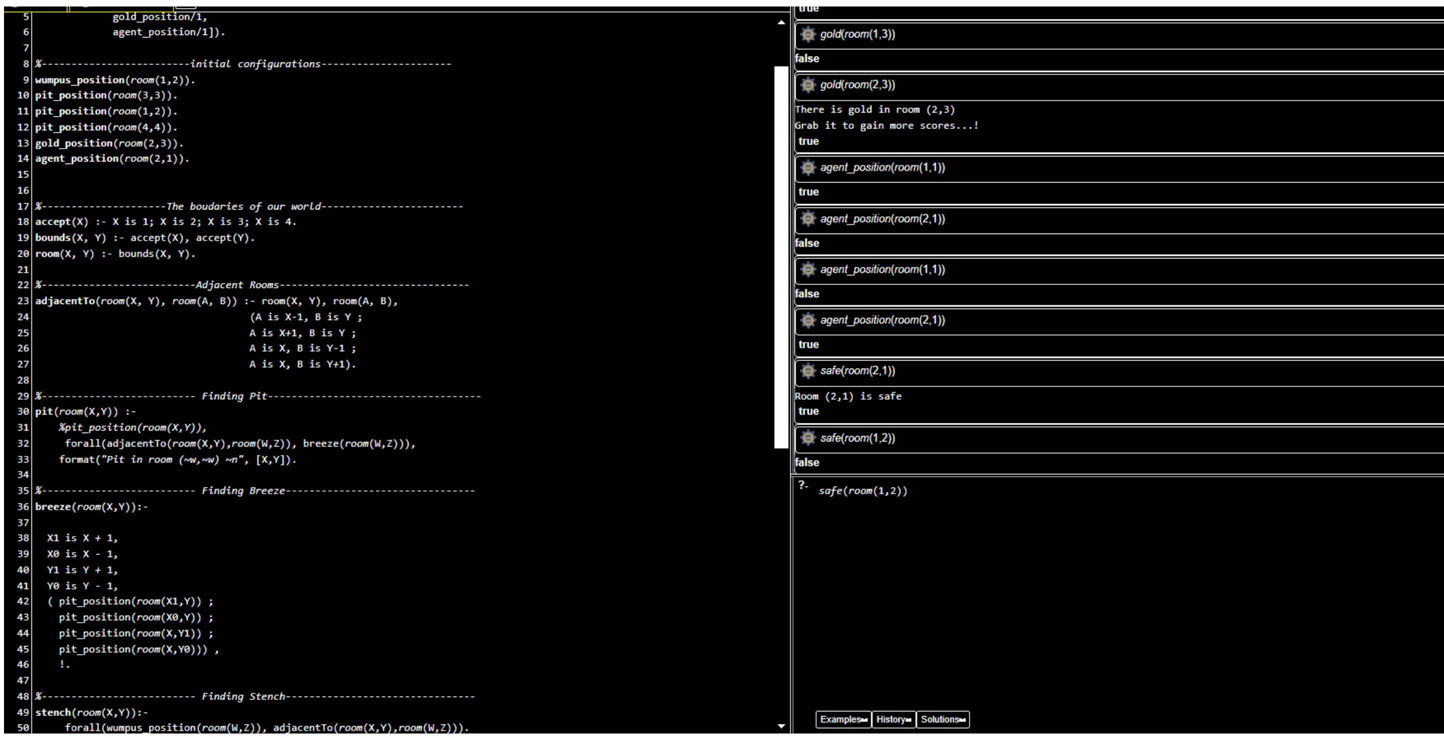
```
%-------------------------initial configurations--------
wumpus_position(room(1,3)).
pit_position(room(3,3)).
pit_position(room(1,2)).
pit_position(room(4,4)).
gold_position(room(4,3)).
agent_position(room(4,3)).
```

The function safe will give as the rooms that are safe (rooms that do not have wumpus and pit), so it will give us true if the rooms do not have wumpus or pit and it will give us false if the rooms have wumpus or pit.



```
  X1 is X + 1,
  X0 is X - 1,
  Y1 is Y + 1,
  Y0 is Y - 1,
  ( pit_position(room(X1,Y)) ;
    pit_position(room(X0,Y)) ;
    pit_position(room(X,Y1)) ;
    pit_position(room(X,Y0))) ,
    !.

%------------------------- Finding Stench------------------------------
stench(room(X,Y)):-
     forall(wumpus_position(room(W,Z)), adjacentTo(room(X,Y),room(W,Z))).

%------------------------- Finding Wumpus------------------------------
wumpus(room(X,Y)):-
    forall(adjacentTo(room(X,Y),room(W,Z)), stench(room(W,Z))),
    format("Wumpus in room (~w,~w) ~n", [X,Y]).

%------------------------Finding Safe Rooms------------------------------
safe(room(W,Z)):-
    \+ wumpus_position(room(W,Z)), \+ pit_position(room(W,Z)),
    format("Room (~w,~w) is safe ~n", [W,Z]).

safe(room(X,Y), room(W,Z)):- %to get the safe positions from a given position

    adjacentTo(room(X,Y) , room(W,Z)) , \+ wumpus_position(room(W,Z)),  \+ pit_position(room(W,Z)).

%------------------------- Shooting The Wumpus------------------------------
shootWumpus(room(X,Y)):-
    adjacentTo(room(X,Y),room(W,Z)) , wumpus_position(room(W,Z)),
    format("The shot has been done succesfully... ~n"),
    format("You Won... ~n").

%------------------------Finding Gold------------------------------
gold(room(X,Y)):-
    gold_position(room(X,Y)),
    format("There is gold in room (~w,~w) ~n", [X,Y]),
    format("Grab it to gain more scores...!").

%------------------------- Grabing The Gold ------------------------------
grabGold(room(X,Y)):-
        agent_position(room(X,Y)), gold_position(room(X,Y)),
        format("You have grabed the gold in room (~w,~w) ~n", [X,Y]),
```

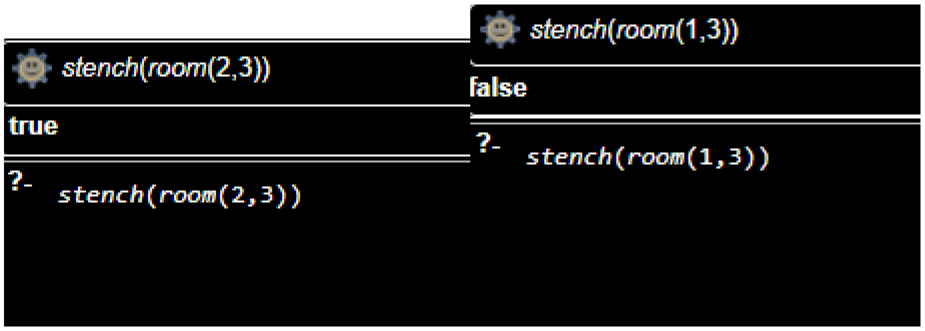
```
There is gold in room (2,3)
Grab it to gain more scores...!
true
gold(room(1,3))
false
gold(room(2,3))
There is gold in room (2,3)
Grab it to gain more scores...!
true
agent_position(room(1,1))
true
agent_position(room(2,1))
false
agent_position(room(1,1))
false
agent_position(room(2,1))
true
safe(room(2,1))
Room (2,1) is safe
true
?- safe(room(2,1))
```



```
            gold_position/1,
            agent_position/1]).

%-----------------------initial configurations---------------------
wumpus_position(room(1,2)).
pit_position(room(3,3)).
pit_position(room(1,2)).
pit_position(room(4,4)).
gold_position(room(2,3)).
agent_position(room(2,1)).


%------------------The boudaries of our world-----------------------
accept(X) :- X is 1; X is 2; X is 3; X is 4.
bounds(X, Y) :- accept(X), accept(Y).
room(X, Y) :- bounds(X, Y).

%------------------------Adjacent Rooms------------------------------
adjacentTo(room(X, Y), room(A, B)) :- room(X, Y), room(A, B),
                                 (A is X-1, B is Y ;
                                  A is X+1, B is Y ;
                                  A is X, B is Y-1 ;
                                  A is X, B is Y+1).

%------------------------- Finding Pit----------------------------------
pit(room(X,Y)) :-
    %pit_position(room(X,Y)),
     forall(adjacentTo(room(X,Y),room(W,Z)), breeze(room(W,Z))),
    format("Pit in room (~w,~w) ~n", [X,Y]).

%------------------------- Finding Breeze------------------------------
breeze(room(X,Y)):-

  X1 is X + 1,
  X0 is X - 1,
  Y1 is Y + 1,
  Y0 is Y - 1,
  ( pit_position(room(X1,Y)) ;
    pit_position(room(X0,Y)) ;
    pit_position(room(X,Y1)) ;
    pit_position(room(X,Y0))) ,
    !.

%------------------------- Finding Stench------------------------------
stench(room(X,Y)):-
     forall(wumpus_position(room(W,Z)), adjacentTo(room(X,Y),room(W,Z))).
```

```
gold(room(1,3))
false
gold(room(2,3))
There is gold in room (2,3)
Grab it to gain more scores...!
true
agent_position(room(1,1))
true
agent_position(room(2,1))
false
agent_position(room(1,1))
false
agent_position(room(2,1))
true
safe(room(2,1))
Room (2,1) is safe
true
safe(room(1,2))
false
?- safe(room(1,2))
```

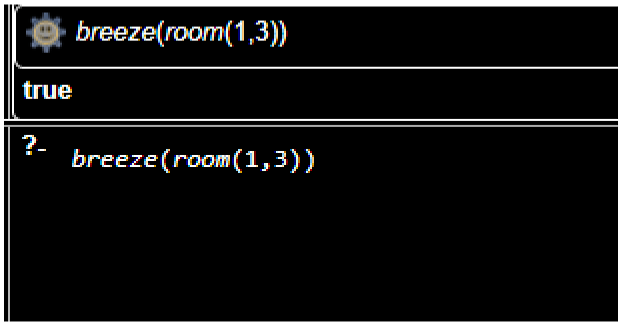
The function stench: we will have a true for the rooms that are adjacent to the wumpus because they are smelly(stench), if the rooms are not adjacent to the wumpus e will get false.



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

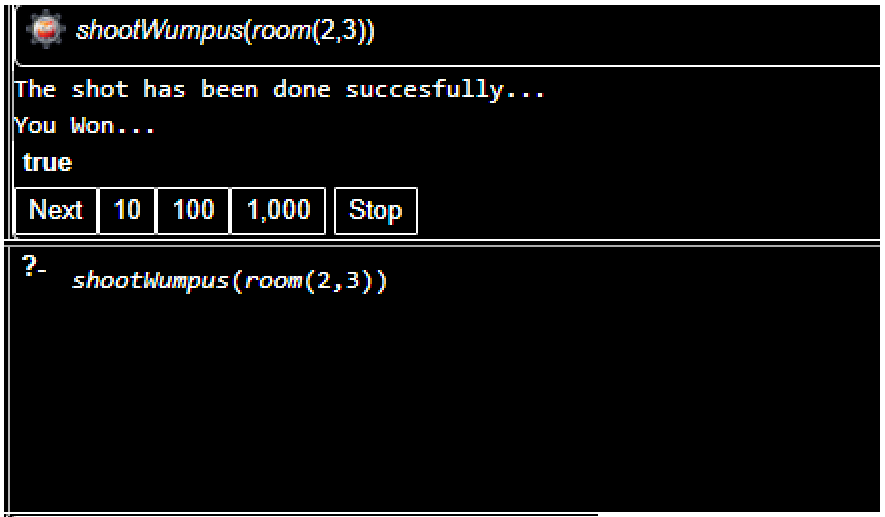
```
stench(room(1,3))
false
?- stench(room(1,3))
```

Breeze: we will have true for the rooms that are adjacent to pit,and false for the rooms that are not adjacent to it.

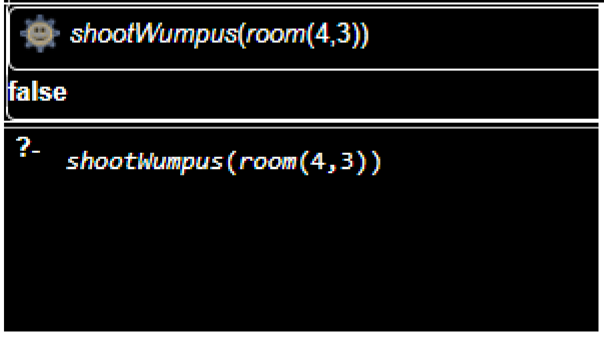


```
breeze(room(1,3))
true
?- breeze(room(1,3))
```

ShootWumpus: if the agent enters the stench room, then he can shoot the Wampus and if he kills him, he will winso will have true if the agent is in the stench room, and false if he is not in the stench room

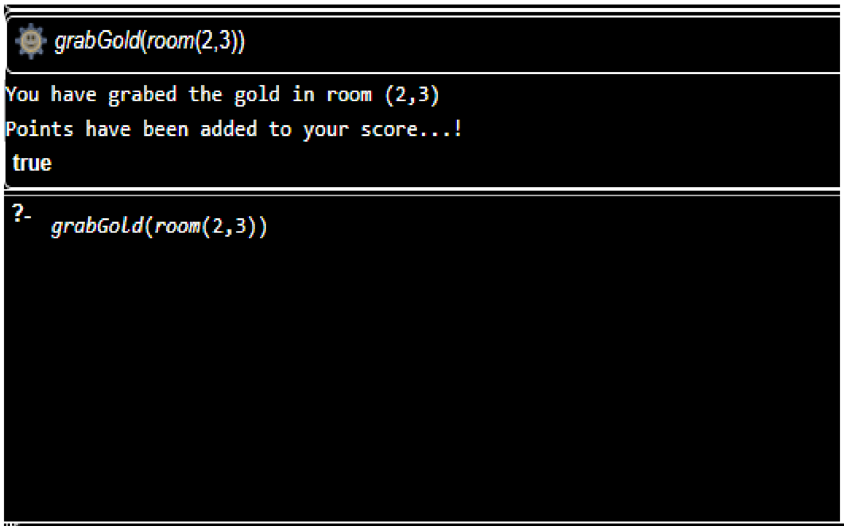


```
shootWumpus(room(2,3))
The shot has been done succesfully...
You Won...
true
?- shootWumpus(room(2,3))
```

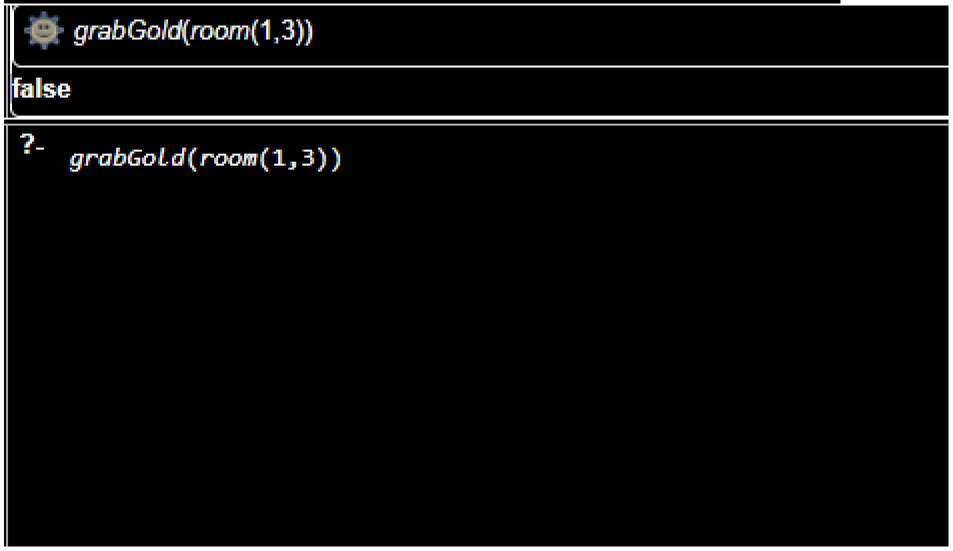


```
shootWumpus(room(4,3))
false
?- shootWumpus(room(4,3))
```

GrabGold: we will get true if the agent is in the gold room, he will grab the gold and he will increase his score.

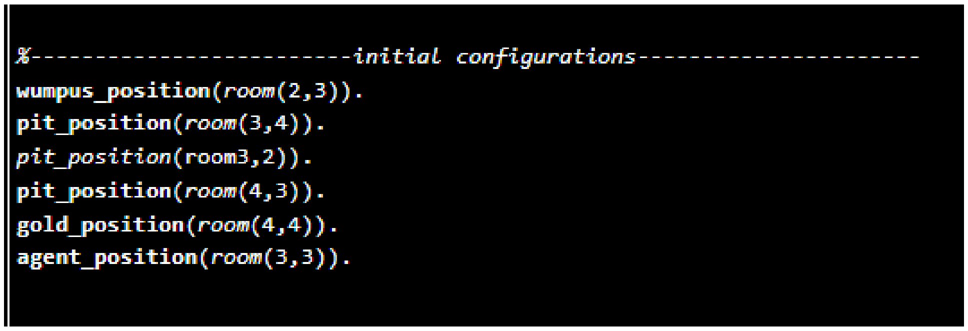


```
grabGold(room(2,3))
You have grabed the gold in room (2,3)
Points have been added to your score...!
true

?- grabGold(room(2,3))
```



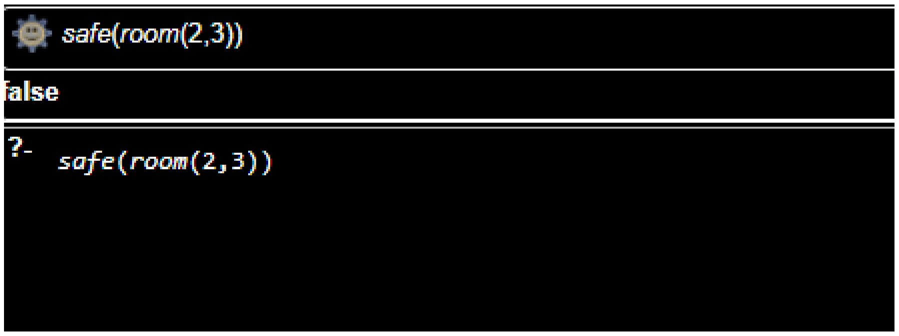
```
grabGold(room(1,3))
false

?- grabGold(room(1,3))
```

Now, we will modify the initial configurations and try again the function:

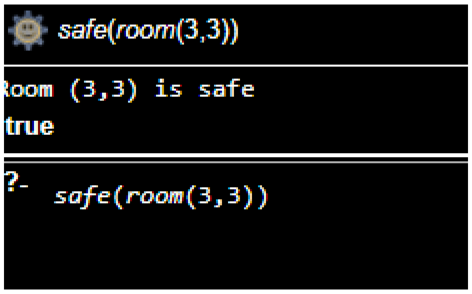


```
%------------------------initial configurations---------------------
wumpus_position(room(2,3)).
pit_position(room(3,4)).
pit_position(room3,2)).
pit_position(room(4,3)).
gold_position(room(4,4)).
agent_position(room(3,3)).
```

Function: safe:

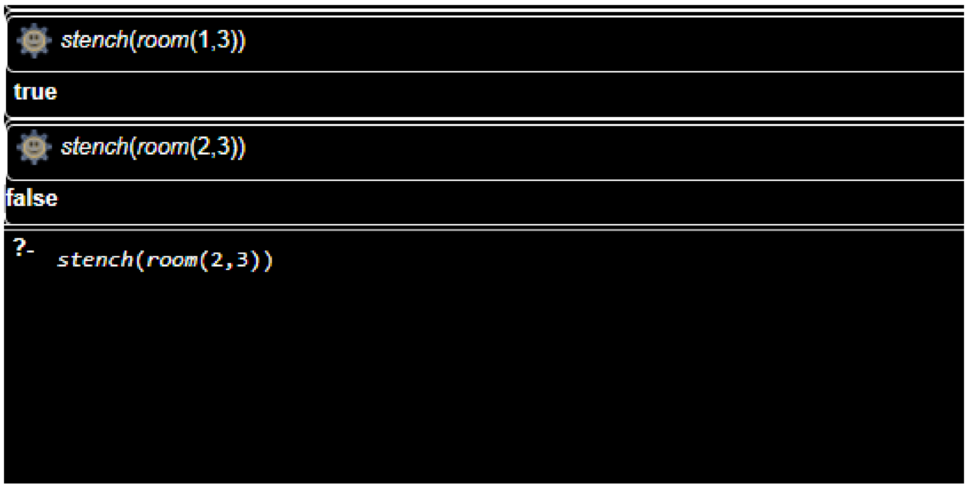


```
safe(room(2,3))
false
?-  safe(room(2,3))
```



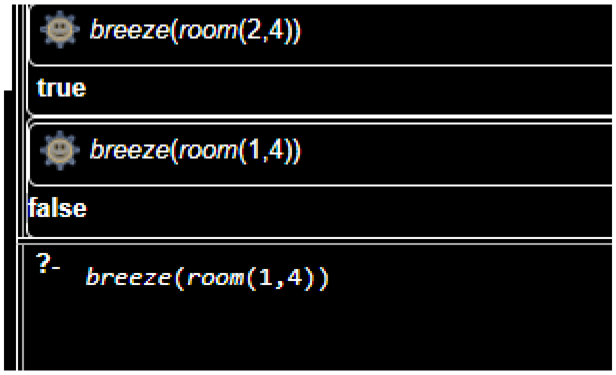
```
safe(room(3,3))
Room (3,3) is safe
true
?-  safe(room(3,3))
```

Function: stench:



```
stench(room(1,3))
true
stench(room(2,3))
false
?-  stench(room(2,3))
```

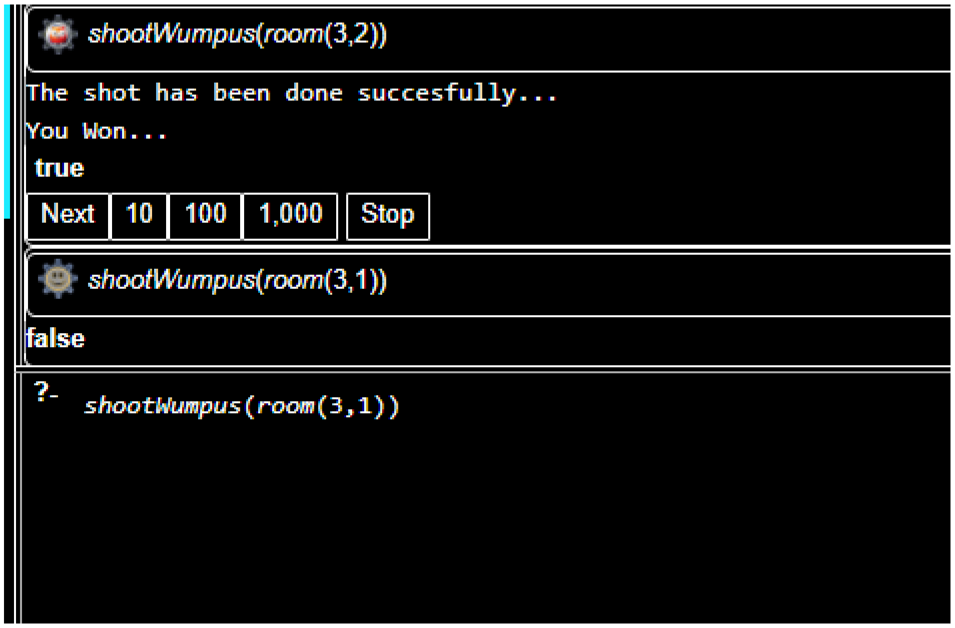
Function: Breeze



```
breeze(room(2,4))
true
breeze(room(1,4))
false
?-  breeze(room(1,4))
```

Now ,we will modify the configuration and try the two functions:grabGold and shootwumpus:

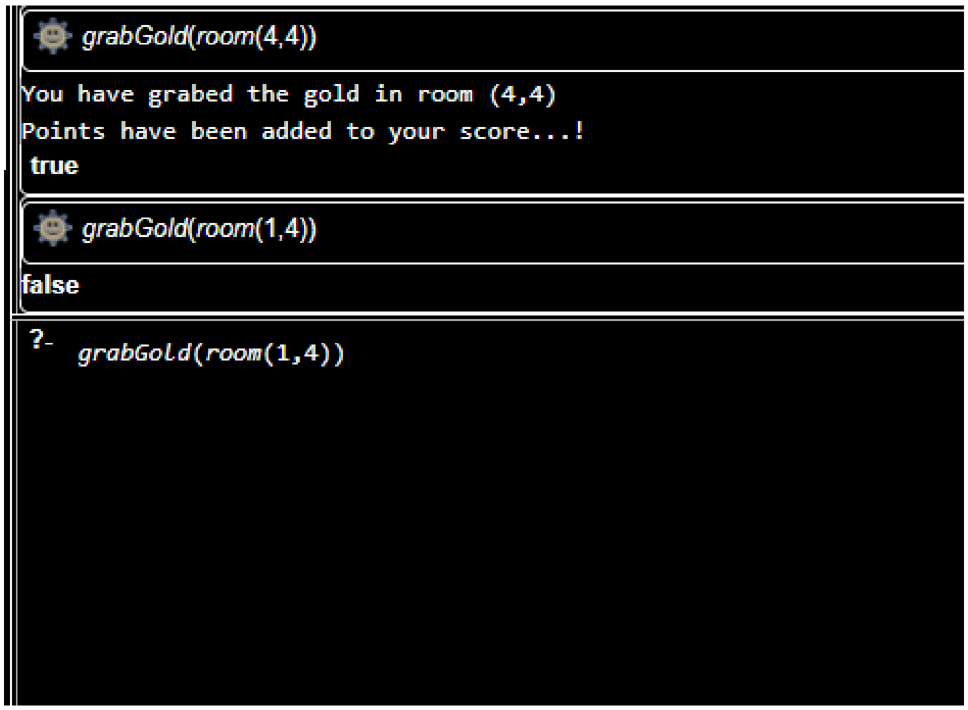
Function:GrabGold:



```
shootWumpus(room(3,2))
The shot has been done succesfully...
You Won...
true
```

```
shootWumpus(room(3,1))
false
?-  shootWumpus(room(3,1))
```

Function:shootwumpus:



```
grabGold(room(4,4))
You have grabed the gold in room (4,4)
Points have been added to your score...!
true

grabGold(room(1,4))
false

?- grabGold(room(1,4))
```

The performance rate:

The situation that impacts the performance rate is when we check the breeze in the pit cell, and it returns false. Users might believe that this cell could be safe for them to move to. However, normally it should output that there no breeze but there is a pit at that room so they should not move to it.

the performance rate:

one situation that impacts the performance, and we have performed 50 queries. So, the performance rate is:

1/50 = 0.02

Limitations and future remedies:

Our program cannot handle the situation when we want to check if there is is a breeze in the room of a pit. This might make the agent follow a wrong path. and to solve this problem if we are in the adjacent rooms to the breeze we need to check if this position contains a pit so that we don't lose the game.