1. Represent the problem of finding the exit of a maze as search space. Describe all 5 components.

    - States:
        - Outside Of the maze (Entrance)
        - Outside Of the maze (Exit)
        - Inside of the maze
            - ViewOnNorth: Agent watching to the North
            - ViewOnSouth; Agent watching to the South
            - ViewOnEast: Agent watching to the East
            - ViewOnWest Agent watching to the West
    - Initial state:
        - Outside Of the maze (Entrance) , Watching to the entrance
    - Actions:
        - Turn Right : The agent turn its axis to the right
        - Turn Left: The agent turn its axis to the left
        - GoForward : The agent goes straight
    - Transition model:
        - (agent.state, turn90º right) -> agent.state=state.right
        - (agent.state, turn90º left) -> agent.state=state.left
        - (agent.state, gofoward) -> agent.position ++
    - Goal test:
        - The agent accessed for the entrance to the maze and came out from the exit
    - Path cost:
        - Time (mins, seconds, ...)
        - Energy Spent

* Considering a generic square maze and simple behaviors

2. Represent a search space (for search algorithms) from your favourite video game/novel/comic/sport, etc... Remember that the search space represents possible states, but it is different from a state machine or an automata.

   Minesweeper

   - States:
     - PlayerAlive
     - PlayerDead
   - Initial state:
     - Alive and All.Mines.Hidden
   - Actions:
     - LeftClickMark(X,Y)
     - RightClickReveal(X,Y)
   - Transition model:
     - (LeftClickMark, (X,Y)) -> mark(X,Y)
     - (RightClickReveal, (X,Y)) -> reveal(X,Y) ->
       - if (X,Y).isMine == True -> PlayerDead
       - if (X,Y).isMine == False -> PlayerAlive
   - Goal test:
     - All the mines are localized and player is Alive
   - Path cost:
     - Time (mins, seconds, ...)

* Simple implementation Minesweeper

does it count as O

-> Discrete
-> Fully Observable
-> KNOWN
-> Static
-> Deterministic
-> Single