CS152 Final Assignment

1. **Problem Definition:**
   As suggested in the possible project ideas list in the assignment description, I focused on developing a program for an agent to navigate in the Wumpus world and so that the agent can build up its KB depending upon its path across the Wumpus World and action decisions. For simplicity, I excluded the arrow throwing feature and also set the goal of winning the Wumpus World as exiting the Wumpus World safely instead of finding the gold and returning back. AI approaches are appropriate for such solving the Wumpus World problem because it requires the agent to partially observe its environment using its sense, stores data on what it sensed whether a stench or a breeze, and use logical reasoning to figure out a path to exit safely. Such knowledged-based agent will be able to solve the problem by updating its KB each time it obtains new observed information, and using propositional logic solving DPLL algorithm to solve the best step to take next step using each updated KB
   As stated in Russel&Norvig, the wumpus environment is "deterministic, discrete, static, single-agent, sequential, and partially observable." Moreover, the PEAS description is as follows:
   - Performance measure:
     - Agent exits safely: 1000 points
     - Agent dies: -1000 points
     - Agent makes one move: -1 point
   - Environment:
     - 4 by 4 grid
     - Agent perceives stench in rooms adjacent to Wumpus
     - Agent perceives breeze in rooms adjacent to pit
     - Initial position is [1,1] and goal position is [4,4]
   - Acutators:
     - Move up
     - Move down
     - Move right
     - Move left
   - Sensors:
     - Breeze
     - Stench
     - Bump (agent hits the boundaries of the wumpus world grid)

2. **Solution Specification:**
   The solution was solved mainly using DPLL algorithm implemented in Python code. The algorithm was divided into mainly two parts: (1) Agent class with functions including initialization of the wumpus world grid, moving in four directions, perceiving breeze or stench etc. and (2) KB class with functions including addition of clauses to the KB, pure symbol heuristic, unit clause heuristic, degree heuristic, and DPLL algorithm. These two

classes enabled the agent to update the KB after each action and search for the next most rational move.

3. **Analysis of Solution:**

Test case 1:

- Initialization:

```
self.__wumpusWorld = [
        ['','','P',''],
        ['','','',''],
        ['W','','',''],
        ['','','','']
```

- Solution:

```
movement: Left, location: [1, 2]
movement: Down, location: [1, 1]
movement: Right, location: [2, 1]
movement: Left, location: [1, 1]
movement: Up, location: [1, 2]
movement: Right, location: [2, 2]
movement: Right, location: [3, 2]
movement: Up, location: [3, 3]
movement: Left, location: [2, 3]
movement: Up, location: [2, 4]
movement: Right, location: [3, 4]
movement: Left, location: [2, 4]
movement: Left, location: [1, 4]
movement: Right, location: [2, 4]
movement: Down, location: [2, 3]
movement: Right, location: [3, 3]
movement: Down, location: [3, 2]
movement: Right, location: [4, 2]
movement: Left, location: [3, 2]
movement: Up, location: [3, 3]
movement: Up, location: [3, 4]
movement: Left, location: [2, 4]
movement: Left, location: [1, 4]
movement: Right, location: [2, 4]
movement: Down, location: [2, 3]
movement: Down, location: [2, 2]
movement: Left, location: [1, 2]
movement: Down, location: [1, 1]
movement: Right, location: [2, 1]
movement: Left, location: [1, 1]
movement: Up, location: [1, 2]
movement: Right, location: [2, 2]
movement: Up, location: [2, 3]
movement: Up, location: [2, 4]
movement: Right, location: [3, 4]
movement: Down, location: [3, 3]
movement: Down, location: [3, 2]
movement: Right, location: [4, 2]
movement: Down, location: [4, 1]
movement: Up, location: [4, 2]
movement: Up, location: [4, 3]
movement: Up, location: [4, 4]
[4, 4] reached. Exiting the Wumpus World.
Total number of times DPLL function is called: 2070
```

- Interpretation: The agent successfully exited the wumpus as it should, but there were many unnecessary moves made in the process, even leading the number of times DPLL function was called to be over 2000. This probably is due to a problem in the function that should move to unvisited rooms. Moreover, it would have been better if I could have represented the path graphically in a grid.

Test case 2

- Initializaton:

```
self.__wumpusWorld = [
        ['','','W',''],
        ['','','','W'],
        ['','','',''],
        ['','','','']
```

- Solution:

```
Action cannot be performed. Agent is DEAD. Location:[4, 2]
Action cannot be performed. Agent is DEAD. Location:[4, 2]
Action cannot be performed. Agent is DEAD. Location:[4, 2]
Action cannot be performed. Agent is DEAD. Location:[4, 2]
Action cannot be performed. Agent is DEAD. Location:[4, 2]
Action cannot be performed. Agent is DEAD. Location:[4, 2]
Action cannot be performed. Agent is DEAD. Location:[4, 2]
Action cannot be performed. Agent is DEAD. Location:[4, 2]
Action cannot be performed. Agent is DEAD. Location:[4, 2]
Action cannot be performed. Agent is DEAD. Location:[4, 2]

---------------------------------------------------------------------------
RecursionError                            Traceback (most recent call last)
<ipython-input-7-9c398fc88168> in <module>
    301
    302 if __name__=='__main__':
--> 303     main()

<ipython-input-7-9c398fc88168> in main()
    295     kb= KnowledgeBase()
```

- Interpretation: Since the two wumpuses are blocking the exit (4,4), it is correct for the solution to output that the agent cannot reach the exit. However, it would have been better if the code does not go into an infinite loop error but prints that the solution is not satisfiable.

4. **References:** I received help from Tuan on coding problems I encountered and also developed on the DPLL algorithm programmed in the last assignment to solve using propositional logic in KB.
5. **Appendices:**
   code for the algorithm:
   https://gist.github.com/leegaeun00/d56dd59958342c68ef3357f2b421eb43