################## Pacman Instructions ####################

Assignment:

Implement

1. Breadth-first search in the class **BFSAgent**,
2. Depth-first search in the class **DFSAgent**,
3. A* search in the class **AStarAgent**,

within the in **pacmanAgents.py** file, using **admissibleHeuristic** as a heuristic function for the **AStarAgent**.

Notes:

- Python 2.7 is required to run the Framework.
- All your code must be inside the **pacmanAgents.py** file.
- **RandomAgent** and **OneStepLookAheadAgent** are implemented as example agents.
- External libraries are not allowed (as you won't submit them).
- Ways to **fail** the assignment if:
    - You try to change any of the system params.
    - Your code didn't run (has errors).
    - You didn't write the code yourself.
    - You submit anything beside **pacmanAgents.py** file.
    - You changed the name of the agent classes.
    - You implemented your own heuristic.
- You are only allowed to use these system functions (accessing/ changing any other functions or variables is considered cheating):
    - **state.getLegalPacmanActions():** return all the legal actions in this state
    - **state.generatePacmanSuccessor(action):** return the next

state if pacman take a certain action (return a new copy, doesn't modify the current state)

- **admissibleHeuristic(state):** estimates the remaining cost from the current state to the goal state
- **state.isWin():** check if this state is win state
- **state.isLose():** check if this state is lose state

- The forward model (**generatePacmanSuccessor**) is limited to a certain amount of calls, don't waste them. If you exceed the limit, **None** will be returned.
- If you did not reach a terminal state, return the action leading to the node with the **minimum total cost**.
- For A* algorithm: $f(n) = g(n) + h(n)$
  - $f(n)$: the total cost of the next node
  - $g(n)$: is the cost of the path since the start node (**in this exercise, this cost is the depth of the current node, i.e. the number of actions from the start till that node**)
  - $h(n)$: is a heuristic function that estimates the remaining cost till the goal (**in this exercise, this heuristic can be calculated using admissibleHeuristic(state) function on the current state**)
- For array sorting, you can use python internal sorting function. example: **array.sort(key=lambda x: admissibleHeuristic(x)).** This example sort the array based on the admissibleHeuristic function.

How to run:

- To play pacman:

```
python pacman.py
```

- To run a certain agent using graphics use the following command:

```
python pacman.py -p AgentName
```