AI – Project 1 Student Cheat Sheet:
Introduction:

- Start early!!!! (Aka start today, now. Even if you just look at the code, do it!)
    - **This project will take a lot of time, and you need to understand the code environment and the lecture material in order to be successful**
    - **So, get help early!**
- The **ONLY** files you will edit are search.py and searchAgents.py
    - Don't change method headers or any other files
    - Do not use other libraries that do not come with the projects as they will not be installed on the online server
- Use the autograder to test your code locally but be aware that **your grade in gradescope is your final grade**.
    - You must test in the online environment as well. Please let us know if you have any issues.
- In this project, we are helping the Pacman find paths through his maze world focusing on search algorithms
- You will need to fill in code at the positions labeled *** YOUR CODE HERE ***
    - You do not need to fill in code anywhere else and should not
- Look at pacman.py
    - Class GameState will give you the abilities of the pacman
- Look at layout.py
    - Class Layout will give you the aspects of the maze

Question 1, 2, 3, 4: Depth First Search, Breath First Search, Uniform Cost Search, and A*

- All functions should be placed in search.py
- Make sure to return a list of actions leading the pacman from the start to the goal
- Usable data structures include Stacks, Queues, and Priority Queues.
    - These are provided in the util.py file
- Each function should keep track of visited states and while determining the next place to check for the destination
- The Algorithms for DFS, BFS, UCS, and A* should be very similar in nature
- For A*:
    - Heuristics take two arguments a state in the search problem and the problem itself
    - Look at the nullHeuristic function shown in search.py for an example
    - How use the Manhattan distance heuristic found in searchagent.py to test your code for A*

Question 5: Finding the Corners

- You will need to complete Question 2 before completing this question
- Find a way to keep track if all of the corner have been reached as this is your goal
    - Your goal is not to reach the destination state
- Do not use a Pacman GameState as a search state
    - All you will need to reference is the start location and the location of the four corners
    - Class Layout will be helpful here

Question 6: Finding the Corners Heuristic

- Remember this function should estimate the cost to a nearest goal (a corner) and this cost should be consistent
    - The autograder will evaluate you on this consistency by checking you code multiple times on different mazes
- Do not forget to keep track of the location of the corners and if a corner has been visited as well as the current cost to the corner

- Hint: An example of the Manhattan distance heuristic is located in util.py and can be used as an example or utilized in your own heuristic
- You should return the total cost of reaching the corner

Question 7: Eating All the Dots

- You will need complete Question 4 before attempting this question
- The goal is to find a way to eat all of the pellets in the maze (not to find the destination state), so do not forget to check for this in your evaluation
    - The solution will depend on the placement of walls, food, and start state
    - The locations of the food can be found in the food grid, which is the second parameter of the state
    - You will not need to deal with any ghosts
- This question’s answer will be based on your implementation of A*

Question 8: Suboptimal Search

- The goal of this question is to create a Pacman that eats the closest dot by finding the path to it
- You will once again want to keep track of the maze’s attributes in some way

Do not forget to fill in the goal test found in AnyFoodSearchProblem

- The goal should be the completion of eating all of the available dots