Pacman competition **Name: Heejong Kim / NetID: hk2451**

I searched some papers related to the best algorithm for Pacman. Most of them are focused on making a better algorithm for 'random' behavior of Ms.pacman ghosts so they used Monte Carlo method. And some of the papers which do not use Monte Carlo were using the evolutionary algorithm. Those methods seem hard to be used if we have a computational limit. And Pacman's ghosts do not have that 'randomness'.
I decided to use A* algorithm and write my own heuristic function for this Pacman competition.

Goal: Eat all pellet and get a high score without bumping into ghosts.

Idea: Use A* search algorithm twice. One for overall game state searches to get the best score and one for searching maze distance to find the shortest path between two positions.

1) First A* search algorithm:
   I used the algorithm to compute **'Reachable maze distance'**. I constructed a reachable maze for computing maze distance using initial positions (pellets, capsules, ghosts, and Pacman). Using the maze information, searched distance between two positions using A* search algorithm.
   Heuristic = [Manhattan distance from current's neighbor position to end position].

2) Second A* search algorithm:
   I used the algorithm to expand Pacman's game state like what we did in homework. In addition to that, I modified some parts along with heuristic function. First, I expanded the node with the game state until it meets the closest pellet. This made my algorithm fast. Second, I used a heuristic function like below:
   *Heuristic = [**Reachable maze distance** from Pacman to the closest pellets] + [Number of pellets left in the maze] + [Weight for losing state].*

Result & Discussion:
I ran 20 times and the average score was 1340.5. Minimum score was 220 and the maximum score was 2070. This algorithm is not a perfect solution but it is a fast and intuitive method.