COEN 166 Artificial Intelligence

Lab Assignment #5 - MultiAgent I

Name: Tanner Kaczmarek ID:00001443463

Problem: Improving the Reflex Agent

Function 1:

```
    def evaluationFunction(self, currentGameState, action):
        """
        Design a better evaluation function here.

        The evaluation function takes in the current and proposed successor
        GameStates (pacman.py) and returns a number, where higher numbers are better.

        The code below extracts some useful information from the state, like the
        remaining food (newFood) and Pacman position after moving (newPos).
        newScaredTimes holds the number of moves that each ghost will remain
        scared because of Pacman having eaten a power pellet.

        Print out these variables to see what you're getting, then combine them
        to create a masterful evaluation function.
        """
        # Useful information you can extract from a GameState (pacman.py)
        successorGameState = currentGameState.generatePacmanSuccessor(action)
        newPos = successorGameState.getPacmanPosition()
        newFood = successorGameState.getFood()
        newGhostStates = successorGameState.getGhostStates()
        newScaredTimes = [ghostState.scaredTimer for ghostState in newGhostStates]

        #print('new Pos: ', newPos, ' new Food: ', newFood, ' newGhostStates: ',
newGhostStates, ' newScaredTimes: ', newScaredTimes, ' \n')

        oldFood = currentGameState.getFood()

        #Finds the closest food
        foods = -1
        rightNextTo = False
        for x in range(newFood.width):
            for y in range(newFood.height):
                if(newFood[x][y] == True):
                    temp = []
                    temp.append(x)
                    temp.append(y)
                    temp2 = manhattanDistance(temp, newPos)
```

```
            if foods < 0 or temp2 < foods:
                foods = temp2
            if oldFood[x][y] == True and newPos[0] == x and newPos[1] == y:
                rightNextTo = True


    #Finds the closest ghost
    ghosts = -1
    x = 0
    while x < len(newGhostStates):
        temp = manhattanDistance(newPos, successorGameState.getGhostPosition(x+1))
        if temp <= 3:
            return temp #ghosts are so close so lets return a super small value
        if ghosts < 0 or temp < ghosts:
            ghosts = temp
        x += 1

    score = successorGameState.getScore()
    if action == Directions.STOP:
        score - 50

    "*** YOUR CODE HERE ***"
    #Food is so close let's just go for that
    if rightNextTo and action != Directions.STOP:
        return successorGameState.getScore() + 100

    #Ghosts are scared, or ghosts are so far away that we can just forget about the ghost
right now
    if newScaredTimes[0] > 0 or ghosts > 10:
        return score - foods*3

    #Ghosts are close and lets calculate a score regarding ghosts and
    return score - foods*3 + ghosts
```

Comment:

The purpose of my evaluation function is to return a score for the next possible state given an action; the score depends on how the successor state correlates to food and ghost proximity and if the ghost can be eaten. A higher score means that the next possible action is a the best one to take.

My evaluation function is revised as follows:

1. The first part of my function is all the values we need from the games state. I added one line to that and got not only the newFood but also the old food. I did this for I could figure out if a food is directly next to my pacman.
2. The next part of my function is getting the distance to the closest food. I used the manhattanDistance function to figure this out. While this is happening I also check if the new

possible action leads to being at a food. To do this I used my oldFood value. With the newFood value, it can be misleading because the newFood will show a false value for the newPos location no matter what, cause even if there was food there before, now there will be no food with the given action.

3. The next part of my function is finding the location of the closest ghost using manhattanDistance function. If the successor state is within a distance of 3 then I will return that distance because that distance is very low. It will lead to other actions being chosen rather than this one.
4. The last part of my function is all my returns besides the one safety return in the closest ghost part of the code. The first thing I do is get the score from the successorGameState. I found out that the best move is hardly ever just standing still so I made stopping a penalty to the high score. The next thing I did was checking that if the food is right next to the pacman, then return a very high value because this should be the best move usually. The next if statement is true if the ghosts can be eaten by pacman, or if the ghosts are so far away they can just be ignored. In that case I take my variable score and subtract it by the closest food. The last part of the code will return with no conditionals. It takes in fact the closest food and closet ghost. There is some priority on going to the closest food over the closest ghost though.