1. **Take the multiagent folder from assignment 2 and copy into a new directory for this practical. We will implement a version of minimax for Ghost Agents to make very smart ghosts.**

Answer:

Main code of Mine:

```
class MinimaxGhost():
    def __init__(self,index,depth = 2,evalFun='scoreEvaluationFunctionGhost'):
        self.index = index
        self.depth = depth
        self.evaluationFunction = util.lookup(evalFun, globals())

    def getAction(self,gameState):
        def Handle(currentgameState, GhostIndex=0, GhostDepth=0):
            if GhostIndex == gameState.getNumAgents():
                GhostIndex = 0
            elif GhostIndex == self.index:
                GhostDepth = GhostDepth+1
            if GhostDepth == self.depth:
                return self.evaluationFunction(currentgameState)
            if currentgameState.isWin():
                return self.evaluationFunction(currentgameState)
            if currentgameState.isLose():
                return self.evaluationFunction(currentgameState)
            if GhostIndex == 0:
                return GhostTier(currentgameState, GhostIndex, GhostDepth, True)
            else:
                return GhostTier(currentgameState, GhostIndex, GhostDepth, False)

        def findminway(glocation1,pclocation):
            import math
            length=math.pow(glocation1[0]-pclocation[0],2)+math.pow(glocation1[1]-pcl
ocation[1],2)
            return length

        def Sorting(LegalActions, index):
            import copy
            import numpy as np
            glocation = copy.deepcopy(gameState.getGhostPosition(index))
            ghostlocation = np.array(glocation)
            pclocation = gameState.getPacmanPosition()
            lenglist = []
            xg = 0
            yg = 0
            mylist  = []
            for action in LegalActions:
                if action == "East":
                    xg = ghostlocation[0]+1
```

```
            elif action == "West":
                xg = ghostlocation[0] - 1
            elif action == "North":
                yg=ghostlocation[1] -1
            elif action == "South":
                yg=ghostlocation[1]+1

            length = findminway([xg,yg],pclocation)
            mylist.append([action,length])
            lenglist.append(length)

        mylist.sort(key=lambda x:x[1])
        indexnum = 0
        for i in mylist:
            LegalActions[indexnum] = i[0]
            indexnum = indexnum+1
        return LegalActions

    def GhostTier(currentGameState, GhostIndex, GhostDepth, Flag):
        Sco = []
        result = []
        Acts = []
        LegalActions = currentGameState.getLegalActions(GhostIndex)
        if len(LegalActions) == 0 or LegalActions == None:
            return self.evaluationFunction(currentGameState)
        if LegalActions:
            LegalActions= Sorting(LegalActions,self.index)

        for action in LegalActions:
            currentState = currentGameState.generateSuccessor(GhostIndex, action)
            currentValue = Handle(currentState, GhostIndex + 1, GhostDepth)
            try:
                if currentValue[1] != None:
                    current_Gamescore = currentValue[1]
            except:
                current_Gamescore = currentValue
            Acts.append(action)
            Sco.append(current_Gamescore)
        if len(Acts) == 0:
            return None
        elif Flag:
            Score = max(Sco)
        else:
            Score = min(Sco)

        index1 = Sco.index(Score)
        result.append(Acts[index1])

        return random.choice(result), Score
```

```
92.
93.         return Handle(gameState, GhostIndex=self.index, GhostDepth=-1)[0]
94.
95.def scoreEvaluationFunctionGhost(currentGameState):
96.
97.    return currentGameState.getScore()
```

2. In this question we will test the new ghostAgent with different pacman agents. In your assignment you were asked to complete a Minimax pacman a version of Expectimax for pacman was provided. If you have not completed the assignment just use the provided expectimax version.

We will perform some experiments to compare the performance of pacman against different types of ghost agent:

a random (not smart) ghost vs a pacman that assumes optimal play from ghosts (ie minimax pacman).

a smart (minmax) ghost vs a pacman that assumes optimal play from ghosts

random ghosts vs pacman that assumes ghosts may not always do an optimal move

smart (minimax) ghosts vs pacman that assumes that ghosts may do suboptimal moves.

Answer:

This results is in a table as the following:

	Adversarial Ghost	**Random Ghost**
Minimax Pacman	Won:0/5 Avg.Score:-506.0	Won:3/5 Avg.Score:92.6
Expectimax Pacman	Won:0/5 Avg.Score:-506.0	Won:2/5 Avg.Score:-112.8

3.Describe the performance (in terms of the distribution) of Pacman in each case.

In which cases is the Pacman agent implementing the correct assumption of the ghosts behaviour?

Answer: We can see the best performance of Pacman is the Pacman with Minimax Agent with the threats of Ghosts under RandomGhost Agent, the second best performance of Pacman is the Pacman with Expectimax Agent with the threats of Ghosts under RandomGhost Agent. And the worst performance of Pacman is the Pacman with the threats of Adversarial Ghosts.Under the threats of Random Ghosts, the Pacman with Minimax Agent implementing the correct assumption of the ghosts behaviour.

4. Describe why the ghosts seem as if they are cooperating when using minimax even though they are not sharing information with each other.

Answer: Because both of them want the Pacman get lower score, and both of them assume the Pacman will move to the best place. So they will get the same information of Pacman in their depth,and they will assume other ghost do the worst choice when they calculate the scores.So the ghosts seem as if they are cooperating when using minimax even though they are not sharing information with each other.