Homework 3: Multi-Agent Search

Part I. Implementation (5%):

- Part 1

```
    # Begin your code (Part 1)
    """
    Implmenting Minmax Tree with recursive function.
    """
    actions = []
    for act in gameState.getLegalActions(0):
        actions.append((act, self.Minimax(gameState.getNextState(0, act), act, self.depth-1, 1, False)))
    action, _ = max(actions, key=lambda x: x[1])
    return action
    # End your code (Part 1)

def Minimax(self, gameState, action, depth, agentid, maxx):
    if gameState.isWin() or gameState.isLose() or (depth == 0 and agentid == 0):
        return self.evaluationFunction(gameState) #Determine terminal state
    actions = []
    if maxx: #max layer
        for action in gameState.getLegalActions(agentid):
            actions.append(self.Minimax(gameState.getNextState(agentid, action), action, depth-1, agentid+1, False))

        return max(actions)
    #min layer
    for action in gameState.getLegalActions(agentid):
        actions.append(self.Minimax(gameState.getNextState(agentid, action), action, depth,
                                    (0 if agentid == gameState.getNumAgents()-1 else agentid+1),
                                    (True if agentid == gameState.getNumAgents()-1 else False)))
    return min(actions)
```

- Part2

```
    # Begin your code (Part 2)
    """
    Maintain (alpha, beta) to represent the upper bound and lower bound of value function.
    This can pruning the min-max tree.
    """
    action = gameState.getLegalActions(0)[0]
    alpha = -self.INF
    beta = self.INF
    for act in gameState.getLegalActions(0):
        v = self.Alpha_beta(gameState.getNextState(0, act), act, self.depth-1, 1, False, alpha, beta)
        if v > alpha:
            alpha = v
            action = act
    return action
    # End your code (Part 2)

def Alpha_beta(self, gameState, action, depth, agentid, maxx, alpha, beta):
    if gameState.isWin() or gameState.isLose() or (depth == 0 and agentid == 0):
        return self.evaluationFunction(gameState) #terminal state
    if maxx:
        v = -self.INF
        for action in gameState.getLegalActions(agentid):
            v = max(v, self.Alpha_beta(gameState.getNextState(agentid, action), action,
                                       depth-1, agentid+1, False, alpha, beta))
            if v > beta:
                break
            alpha = max(alpha, v)
        return v
    #if min
    v = self.INF
    for action in gameState.getLegalActions(agentid):
        v = min(v, self.Alpha_beta(gameState.getNextState(agentid, action), action, depth,
                                   agentid=(0 if agentid == gameState.getNumAgents()-1 else agentid+1),
                                   maxx=(True if agentid == gameState.getNumAgents()-1 else False), alpha=alpha, beta=beta))
        if v < alpha:
            break
        beta = min(beta, v)
    return v
```

- Part3

```
    # Begin your code (Part 3)
    """
    The different part of expectimax and minmax agent is the min layer.
    The min part take the mean value of the child state values.
    """
    actions = []
    for act in gameState.getLegalActions(0):
        next_state = gameState.getNextState(0, act)
        actions.append((act, self.Expectimax(next_state, act, self.depth-1, 1, False)))
    action = ''
    best_v = -10000000
    for act, v in actions:
        if best_v < v and act != 'Stop':
            best_v = v
            action = act
    #action, _ = max(actions, key=lambda x: x[1])
    #print(best_v)
    return action
    # End your code (Part 3)

def Expectimax(self, gameState, action, depth, agentid, maxx):
    if gameState.isWin() or gameState.isLose() or (depth == 0 and agentid == 0):
        return self.evaluationFunction(gameState)*(depth+1)
    actions = gameState.getLegalActions(agentid)
    if maxx:
        values = []
        for action in actions:
            values.append(self.Expectimax(gameState.getNextState(agentid, action), action, depth-1, agentid+1, False))
        return max(values)
    value = 0
    for action in gameState.getLegalActions(agentid):
        value += self.Expectimax(gameState.getNextState(agentid, action), action, depth,
                                 (0 if agentid == gameState.getNumAgents()-1 else agentid+1),
                                 (True if agentid == gameState.getNumAgents()-1 else False))
        # Take mean of values

    return value/len(actions)
```

- Part4

```
"""
Since eating scared ghost can get 200 points.
My poolicy is first find capsule and turn the ghost into scare and eat it.
When the capsule is gone, the agent then eat all the food and hide from ghost.
When the ghost is scared, the value is high when close to ghost
otherwise the value is very low when closing to ghost(near to lose).
The distance  between agent and food and ghost is computed by bfs.
"""
```

```
# Begin your code (Part 4)
def BFS2(xy1, xy2):
    q = Queue()
    vis = {}
    q.push(xy1)
    vis[xy1] = 0
    while not q.isEmpty():
        pos = q.pop()
        if pos == xy2:
            return vis[pos]
        if not currentGameState.hasWall(pos[0] + 1,pos[1]) and (pos[0] + 1,pos[1]) not in vis:
            q.push((pos[0] + 1,pos[1]))
            vis[(pos[0] + 1,pos[1])] = vis[pos] + 1
        if not currentGameState.hasWall(pos[0],pos[1] + 1) and (pos[0],pos[1] + 1) not in vis:
            q.p
            vis                                                    1
        if not currentGameState.hasWall(pos[0] - 1,pos[1]) and (pos[0] - 1,pos[1]) not in vis:
            q.push((pos[0] - 1,pos[1]))
            vis[(pos[0] - 1,pos[1])] = vis[pos] + 1
        if not currentGameState.hasWall(pos[0],pos[1] - 1) and (pos[0],pos[1] - 1) not in vis:
            q.push((pos[0],pos[1] - 1))
            vis[(pos[0],pos[1] - 1)] = vis[pos] + 1
    return None
```

(parameter) currentGameState: Any

```
def BFS(xy1):
    q = Queue()
    vis = {}
    q.push(xy1)
    vis[xy1] = 0
    while not q.isEmpty():
        pos = q.pop()
        if currentGameState.hasFood(pos[0], pos[1]):
            return vis[pos]
        if not currentGameState.hasWall(pos[0] + 1,pos[1]) and (pos[0] + 1,pos[1]) not in vis:
            q.push((pos[0] + 1,pos[1]))
            vis[(pos[0] + 1,pos[1])] = vis[pos] + 1
        if not currentGameState.hasWall(pos[0],pos[1] + 1) and (pos[0],pos[1] + 1) not in vis:
            q.push((pos[0],pos[1] + 1))
            vis[(pos[0],pos[1] + 1)] = vis[pos] + 1
        if not currentGameState.hasWall(pos[0] - 1,pos[1]) and (pos[0] - 1,pos[1]) not in vis:
            q.push((pos[0] - 1,pos[1]))
            vis[(pos[0] - 1,pos[1])] = vis[pos] + 1
        if not currentGameState.hasWall(pos[0],pos[1] - 1) and (pos[0],pos[1] - 1) not in vis:
            q.push((pos[0],pos[1] - 1))
            vis[(pos[0],pos[1] - 1)] = vis[pos] + 1
    return None
```

```
if currentGameState.isLose():
    return -10000000

score = currentGameState.getScore()
GhostStates = currentGameState.getGhostStates()
pos = currentGameState.getPacmanPosition()
ScaredGhosttime = 0
ScaredGhostdis = 1000000
for state in GhostStates:
    dis = BFS2(pos, state.getPosition())
    if dis is not None:
        ScaredGhosttime += state.scaredTimer
        ScaredGhostdis = min(dis,ScaredGhostdis)
nearestFoodDistance = BFS(pos)
value = score
if ScaredGhosttime > 2 and dis > 0:
    value += 250/ScaredGhostdis
if len(currentGameState.getCapsules()):
    NearestCapsuleDistance = min([BFS2(pos, cap) for cap in currentGameState.getCapsules()])
    value += 10/NearestCapsuleDistance+10
if nearestFoodDistance is not None:
    value += 10/nearestFoodDistance+5

return value
# End your code (Part 4)
```

Part II. Results & Analysis (5%):

- For Part4: Initially I want to use evaluation function to directly control the agent, but I fin d it is hard because the tree search. Since the expectimax agent will evaluate the future of two step actions, so the agent would do next thing without reach the current goal. For exa mple, the agent is going to eat scared ghost and it then go to eat food around when it appr oach ghost, since it consider the goal of eating ghost is accomplished. I solve it with some time. I think the one step look expctimax would not have this problem. I can design the ev aluation function as state-action Q function and the agent can choose the action with max value.

```
Question part1
==============

*** PASS: test_cases/part1/0-eval-function-lose-states-1.test
*** PASS: test_cases/part1/0-eval-function-lose-states-2.test
*** PASS: test_cases/part1/0-eval-function-win-states-1.test
*** PASS: test_cases/part1/0-eval-function-win-states-2.test
*** PASS: test_cases/part1/0-lecture-6-tree.test
*** PASS: test_cases/part1/0-small-tree.test
*** PASS: test_cases/part1/1-1-minmax.test
*** PASS: test_cases/part1/1-2-minmax.test
*** PASS: test_cases/part1/1-3-minmax.test
*** PASS: test_cases/part1/1-4-minmax.test
*** PASS: test_cases/part1/1-5-minmax.test
*** PASS: test_cases/part1/1-6-minmax.test
*** PASS: test_cases/part1/1-7-minmax.test
*** PASS: test_cases/part1/1-8-minmax.test
*** PASS: test_cases/part1/2-1a-vary-depth.test
*** PASS: test_cases/part1/2-1b-vary-depth.test
*** PASS: test_cases/part1/2-2a-vary-depth.test
*** PASS: test_cases/part1/2-2b-vary-depth.test
*** PASS: test_cases/part1/2-3a-vary-depth.test
*** PASS: test_cases/part1/2-3b-vary-depth.test
*** PASS: test_cases/part1/2-4a-vary-depth.test
*** PASS: test_cases/part1/2-4b-vary-depth.test
*** PASS: test_cases/part1/2-one-ghost-3level.test
*** PASS: test_cases/part1/3-one-ghost-4level.test
*** PASS: test_cases/part1/4-two-ghosts-3level.test
*** PASS: test_cases/part1/5-two-ghosts-4level.test
*** PASS: test_cases/part1/6-tied-root.test
*** PASS: test_cases/part1/7-1a-check-depth-one-ghost.test
*** PASS: test_cases/part1/7-1b-check-depth-one-ghost.test
*** PASS: test_cases/part1/7-1c-check-depth-one-ghost.test
*** PASS: test_cases/part1/7-2a-check-depth-two-ghosts.test
*** PASS: test_cases/part1/7-2b-check-depth-two-ghosts.test
*** PASS: test_cases/part1/7-2c-check-depth-two-ghosts.test
*** Running MinimaxAgent on smallClassic 1 time(s).
Pacman died! Score: 84
Average Score: 84.0
Scores:        84.0
Win Rate:      0/1 (0.00)
Record:        Loss
*** Finished running MinimaxAgent on smallClassic after 1 seconds.
*** Won 0 out of 1 games. Average score: 84.000000 ***
*** PASS: test_cases/part1/8-pacman-game.test

### Question part1: 20/20 ###
```

```
Question part2
==============

*** PASS: test_cases/part2/0-eval-function-lose-states-1.test
*** PASS: test_cases/part2/0-eval-function-lose-states-2.test
*** PASS: test_cases/part2/0-eval-function-win-states-1.test
*** PASS: test_cases/part2/0-eval-function-win-states-2.test
*** PASS: test_cases/part2/0-lecture-6-tree.test
*** PASS: test_cases/part2/0-small-tree.test
*** PASS: test_cases/part2/1-1-minmax.test
*** PASS: test_cases/part2/1-2-minmax.test
*** PASS: test_cases/part2/1-3-minmax.test
*** PASS: test_cases/part2/1-4-minmax.test
*** PASS: test_cases/part2/1-5-minmax.test
*** PASS: test_cases/part2/1-6-minmax.test
*** PASS: test_cases/part2/1-7-minmax.test
*** PASS: test_cases/part2/1-8-minmax.test
*** PASS: test_cases/part2/2-1a-vary-depth.test
*** PASS: test_cases/part2/2-1b-vary-depth.test
*** PASS: test_cases/part2/2-2a-vary-depth.test
*** PASS: test_cases/part2/2-2b-vary-depth.test
*** PASS: test_cases/part2/2-3a-vary-depth.test
*** PASS: test_cases/part2/2-3b-vary-depth.test
*** PASS: test_cases/part2/2-4a-vary-depth.test
*** PASS: test_cases/part2/2-4b-vary-depth.test
*** PASS: test_cases/part2/2-one-ghost-3level.test
*** PASS: test_cases/part2/3-one-ghost-4level.test
*** PASS: test_cases/part2/4-two-ghosts-3level.test
*** PASS: test_cases/part2/5-two-ghosts-4level.test
*** PASS: test_cases/part2/6-tied-root.test
*** PASS: test_cases/part2/7-1a-check-depth-one-ghost.test
*** PASS: test_cases/part2/7-1b-check-depth-one-ghost.test
*** PASS: test_cases/part2/7-1c-check-depth-one-ghost.test
*** PASS: test_cases/part2/7-2a-check-depth-two-ghosts.test
*** PASS: test_cases/part2/7-2b-check-depth-two-ghosts.test
*** PASS: test_cases/part2/7-2c-check-depth-two-ghosts.test
*** Running AlphaBetaAgent on smallClassic 1 time(s).
Pacman died! Score: 84
Average Score: 84.0
Scores:        84.0
Win Rate:      0/1 (0.00)
Record:        Loss
*** Finished running AlphaBetaAgent on smallClassic after 0 seconds.
*** Won 0 out of 1 games. Average score: 84.000000 ***
*** PASS: test_cases/part2/8-pacman-game.test

### Question part2: 25/25 ###
```

```
Question part3
==============

*** PASS: test_cases/part3/0-eval-function-lose-states-1.test
*** PASS: test_cases/part3/0-eval-function-lose-states-2.test
*** PASS: test_cases/part3/0-eval-function-win-states-1.test
*** PASS: test_cases/part3/0-eval-function-win-states-2.test
*** PASS: test_cases/part3/0-expectimax1.test
*** PASS: test_cases/part3/1-expectimax2.test
*** PASS: test_cases/part3/2-one-ghost-3level.test
*** PASS: test_cases/part3/3-one-ghost-4level.test
*** PASS: test_cases/part3/4-two-ghosts-3level.test
*** PASS: test_cases/part3/5-two-ghosts-4level.test
*** PASS: test_cases/part3/6-1a-check-depth-one-ghost.test
*** PASS: test_cases/part3/6-1b-check-depth-one-ghost.test
*** PASS: test_cases/part3/6-1c-check-depth-one-ghost.test
*** PASS: test_cases/part3/6-2a-check-depth-two-ghosts.test
*** PASS: test_cases/part3/6-2b-check-depth-two-ghosts.test
*** PASS: test_cases/part3/6-2c-check-depth-two-ghosts.test
*** Running ExpectimaxAgent on smallClassic 1 time(s).
Pacman died! Score: 84
Average Score: 84.0
Scores:        84.0
Win Rate:      0/1 (0.00)
Record:        Loss
*** Finished running ExpectimaxAgent on smallClassic after 0 seconds.
*** Won 0 out of 1 games. Average score: 84.000000 ***
*** PASS: test_cases/part3/7-pacman-game.test

### Question part3: 25/25 ###
```

```
Question part4
==============

Pacman emerges victorious! Score: 1346
Pacman emerges victorious! Score: 1372
Pacman emerges victorious! Score: 1319
Pacman emerges victorious! Score: 1359
Pacman emerges victorious! Score: 1328
Pacman emerges victorious! Score: 1368
Pacman emerges victorious! Score: 1367
Pacman emerges victorious! Score: 1362
Pacman emerges victorious! Score: 1142
Pacman emerges victorious! Score: 1366
Average Score: 1332.9
Scores:        1346.0, 1372.0, 1319.0, 1359.0, 1328.0, 1368.0, 1367.0, 1362.0, 1142.0, 1366.0
Win Rate:      10/10 (1.00)
Record:        Win, Win, Win, Win, Win, Win, Win, Win, Win, Win
*** PASS: test_cases/part4/grade-agent.test (8 of 8 points)
*** EXTRA CREDIT: 2 points
***     1332.9 average score (4 of 4 points)
***         Grading scheme:
***          < 500:  0 points
***         >= 500:  2 points
***         >= 1000:  4 points
***     10 games not timed out (2 of 2 points)
***         Grading scheme:
***          < 0:  fail
***         >= 0:  0 points
***         >= 5:  1 points
***         >= 10:  2 points
***     10 wins (4 of 4 points)
***         Grading scheme:
***          < 1:  fail
***         >= 1:  1 points
***         >= 4:  2 points
***         >= 7:  3 points
***         >= 10:  4 points

### Question part4: 10/10 ###
```

```
Provisional grades
==================
Question part1: 20/20
Question part2: 25/25
Question part3: 25/25
Question part4: 10/10
------------------
Total: 80/80


                ALL HAIL GRANDPAC.
           LONG LIVE THE GHOSTBUSTING KING.

                 ---      ----      ---
                 |  \    /  + \    /  |
                 | + \--/      \--/ + |
                 |   +     +          |
                 | +     +        +   |
                @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
              @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
            @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
            @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
            \   @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
             \ /  @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
             V   \   @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                  \ /  @@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                   V     @@@@@@@@@@@@@@@@@@@@@@@@@@
                           @@@@@@@@@@@@@@@@@@@@@@@@
                         /\      @@@@@@@@@@@@@@@@@@@@@@@
                        /  \   @@@@@@@@@@@@@@@@@@@@@@@@@@@
                    /\  /    @@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                   /  \ @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                  /    @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                   @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                     @@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                       @@@@@@@@@@@@@@@@@@@@@@@@
                         @@@@@@@@@@@@@@@@@@@@
```