**Marks from the Auto Grader:**

**Q1: 4/4**

**Q2: 5/5**

**Q3: 5/5**

**Q4: 5/5**

**Q5: 6/6**

**Total: 25/25**

**Explanations:**

**Q1:** For this question, not many lines of code was required. The problem was to make sure that Pac-man gets to eat all the food there and avoid the enemy present in the grid. Here, I start by retrieving the food location using the inbuilt function: ‘getFood()’. Also, I have retreived the current location of Pac-man, using getPacmanPosition() inbuilt function. Initially, the distance from the food is set as –infinity using the python function ,’*float(-inf)*’. If the action sent to the function is STOP, the we return the –infinity which denoted no other moves. Now to make sure that Pacman is not near or about to die by the enemy, a for loop is being run, and we use the if loop: if the position of the enemy is same as the variable ‘here’ (which has the position of pacman) and if the ghost is not white/can kill you, it returns float(-inf). After avoiding the enemy, the main for loop that gives the food position of the nearest food item present and the the min distance is returned.

**Q2:** Here, I have defined three main functions that willbe used tocreate the entire graph. In any minimax problem, a graph containing the alternate levels of moves for the player and the opponent is used. Hence, I have defined a main function which initializes the graph, min\_max(). It takes the game state and the agentcounter as the parameters. In the first if loop, we check if the depth of the current node is equal to the game depth or if it is a winning state or a losing state, we call the predefined evaluation function since there is nothing else we can do. Now, if the agentcounter is ==0 we call the find\_max\_value() function else call the find\_minimum\_value() function. The actionList variable contains the next possible move. In the find max value function I have initialized the max as –infinity and compare the upcoming numbers with it. In the following for loop, if the current position has a single value then we have found the next move and we can compare it with the current max value to update the ma value. Or else, we take the ‘1’ indexed value. In this for loop we again call the main min\_max function and that function may call the find min value function. This leads to generation of MIN->MAX->MIN->MAX type of pattern. The find min value function is the same as find max function except that min is initialized by +infinity.

**Q3:** The alpha beta pruning is very similar to minimax algorithm. Hence, the main min\_max function is the same. Apart from this the find\_max and find\_min function is also very similar to minimax. ‘-float(inf)’ is for max and ‘float(inf)’ is for min. Here there are extra values which has to be compared, which are alpha -> (a) and beta -> (b). Hence, after comparing it with maximum, we compare if the value is greater than b (in finding max), then we return the next value. Similarly, in find\_min we compare with x or a (BOTH VARIABLES ARE THE SAME HERE), and return the min movement and next\_value.

**Q4:** For this, I have used to same functions as defined in the above two questions except the find\_min\_value function. Instead of that, I have used the find\_e functions where the probability of expecti node is to be found. The find\_max() and expect\_max() has the same functionality. In the find max value function I have initialized the max as –infinity and compare the upcoming numbers with it. In the following for loop, if the current position has a single value then we have found the next move and we can compare it with the current max value to update the ma value. Similarly, the for loop for expectimax does the same this, only last return logic is different. The enemy\_action is stored in max\_e[0] and the prob is multiplied by the next value (prob \* next\_value) and it is stored in max\_e[1].

**Q5:** For this question we had to code a betterEvaluationFunction. For that I took the current position, as a list, in the variable called ‘now\_state’ and the food positions in the variable ‘Food\_position’. Further, for a loop is run to calculate the Manhattan distance between the current position and each food item. Each are multiplied by -1 and hence at the end, we just send the addition of the current score and the max value (max(food\_items)). Also, if there are not food items present then we append 0 and hence that means that the current score cannot be increased.

**Auto Grader output:**

Question q1

===========

Pacman emerges victorious! Score: 1246

Pacman emerges victorious! Score: 1237

Pacman emerges victorious! Score: 1235

Pacman emerges victorious! Score: 1245

Pacman emerges victorious! Score: 1233

Pacman emerges victorious! Score: 1231

Pacman emerges victorious! Score: 1235

Pacman emerges victorious! Score: 1253

Pacman emerges victorious! Score: 1231

Pacman emerges victorious! Score: 1236

Average Score: 1238.2

Scores: 1246.0, 1237.0, 1235.0, 1245.0, 1233.0, 1231.0, 1235.0, 1253.0, 1231.0, 1236.0

Win Rate: 10/10 (1.00)

Record: Win, Win, Win, Win, Win, Win, Win, Win, Win, Win

\*\*\* PASS: test\_cases\q1\grade-agent.test (4 of 4 points)

\*\*\* 1238.2 average score (2 of 2 points)

\*\*\* Grading scheme:

\*\*\* < 500: 0 points

\*\*\* >= 500: 1 points

\*\*\* >= 1000: 2 points

\*\*\* 10 games not timed out (0 of 0 points)

\*\*\* Grading scheme:

\*\*\* < 10: fail

\*\*\* >= 10: 0 points

\*\*\* 10 wins (2 of 2 points)

\*\*\* Grading scheme:

\*\*\* < 1: fail

\*\*\* >= 1: 0 points

\*\*\* >= 5: 1 points

\*\*\* >= 10: 2 points

### Question q1: 4/4 ###

Question q2

===========

\*\*\* PASS: test\_cases\q2\0-lecture-6-tree.test

\*\*\* PASS: test\_cases\q2\0-small-tree.test

\*\*\* PASS: test\_cases\q2\1-1-minmax.test

\*\*\* PASS: test\_cases\q2\1-2-minmax.test

\*\*\* PASS: test\_cases\q2\1-3-minmax.test

\*\*\* PASS: test\_cases\q2\1-4-minmax.test

\*\*\* PASS: test\_cases\q2\1-5-minmax.test

\*\*\* PASS: test\_cases\q2\1-6-minmax.test

\*\*\* PASS: test\_cases\q2\1-7-minmax.test

\*\*\* PASS: test\_cases\q2\1-8-minmax.test

\*\*\* PASS: test\_cases\q2\2-1a-vary-depth.test

\*\*\* PASS: test\_cases\q2\2-1b-vary-depth.test

\*\*\* PASS: test\_cases\q2\2-2a-vary-depth.test

\*\*\* PASS: test\_cases\q2\2-2b-vary-depth.test

\*\*\* PASS: test\_cases\q2\2-3a-vary-depth.test

\*\*\* PASS: test\_cases\q2\2-3b-vary-depth.test

\*\*\* PASS: test\_cases\q2\2-4a-vary-depth.test

\*\*\* PASS: test\_cases\q2\2-4b-vary-depth.test

\*\*\* PASS: test\_cases\q2\2-one-ghost-3level.test

\*\*\* PASS: test\_cases\q2\3-one-ghost-4level.test

\*\*\* PASS: test\_cases\q2\4-two-ghosts-3level.test

\*\*\* PASS: test\_cases\q2\5-two-ghosts-4level.test

\*\*\* PASS: test\_cases\q2\6-tied-root.test

\*\*\* PASS: test\_cases\q2\7-1a-check-depth-one-ghost.test

\*\*\* PASS: test\_cases\q2\7-1b-check-depth-one-ghost.test

\*\*\* PASS: test\_cases\q2\7-1c-check-depth-one-ghost.test

\*\*\* PASS: test\_cases\q2\7-2a-check-depth-two-ghosts.test

\*\*\* PASS: test\_cases\q2\7-2b-check-depth-two-ghosts.test

\*\*\* PASS: test\_cases\q2\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 2 seconds.

\*\*\* Won 0 out of 1 games. Average score: 84.000000 \*\*\*

\*\*\* PASS: test\_cases\q2\8-pacman-game.test

### Question q2: 5/5 ###

Question q3

===========

\*\*\* PASS: test\_cases\q3\0-lecture-6-tree.test

\*\*\* PASS: test\_cases\q3\0-small-tree.test

\*\*\* PASS: test\_cases\q3\1-1-minmax.test

\*\*\* PASS: test\_cases\q3\1-2-minmax.test

\*\*\* PASS: test\_cases\q3\1-3-minmax.test

\*\*\* PASS: test\_cases\q3\1-4-minmax.test

\*\*\* PASS: test\_cases\q3\1-5-minmax.test

\*\*\* PASS: test\_cases\q3\1-6-minmax.test

\*\*\* PASS: test\_cases\q3\1-7-minmax.test

\*\*\* PASS: test\_cases\q3\1-8-minmax.test

\*\*\* PASS: test\_cases\q3\2-1a-vary-depth.test

\*\*\* PASS: test\_cases\q3\2-1b-vary-depth.test

\*\*\* PASS: test\_cases\q3\2-2a-vary-depth.test

\*\*\* PASS: test\_cases\q3\2-2b-vary-depth.test

\*\*\* PASS: test\_cases\q3\2-3a-vary-depth.test

\*\*\* PASS: test\_cases\q3\2-3b-vary-depth.test

\*\*\* PASS: test\_cases\q3\2-4a-vary-depth.test

\*\*\* PASS: test\_cases\q3\2-4b-vary-depth.test

\*\*\* PASS: test\_cases\q3\2-one-ghost-3level.test

\*\*\* PASS: test\_cases\q3\3-one-ghost-4level.test

\*\*\* PASS: test\_cases\q3\4-two-ghosts-3level.test

\*\*\* PASS: test\_cases\q3\5-two-ghosts-4level.test

\*\*\* PASS: test\_cases\q3\6-tied-root.test

\*\*\* PASS: test\_cases\q3\7-1a-check-depth-one-ghost.test

\*\*\* PASS: test\_cases\q3\7-1b-check-depth-one-ghost.test

\*\*\* PASS: test\_cases\q3\7-1c-check-depth-one-ghost.test

\*\*\* PASS: test\_cases\q3\7-2a-check-depth-two-ghosts.test

\*\*\* PASS: test\_cases\q3\7-2b-check-depth-two-ghosts.test

\*\*\* PASS: test\_cases\q3\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 2 seconds.

\*\*\* Won 0 out of 1 games. Average score: 84.000000 \*\*\*

\*\*\* PASS: test\_cases\q3\8-pacman-game.test

### Question q3: 5/5 ###

Question q4

===========

\*\*\* PASS: test\_cases\q4\0-expectimax1.test

\*\*\* PASS: test\_cases\q4\1-expectimax2.test

\*\*\* PASS: test\_cases\q4\2-one-ghost-3level.test

\*\*\* PASS: test\_cases\q4\3-one-ghost-4level.test

\*\*\* PASS: test\_cases\q4\4-two-ghosts-3level.test

\*\*\* PASS: test\_cases\q4\5-two-ghosts-4level.test

\*\*\* PASS: test\_cases\q4\6-1a-check-depth-one-ghost.test

\*\*\* PASS: test\_cases\q4\6-1b-check-depth-one-ghost.test

\*\*\* PASS: test\_cases\q4\6-1c-check-depth-one-ghost.test

\*\*\* PASS: test\_cases\q4\6-2a-check-depth-two-ghosts.test

\*\*\* PASS: test\_cases\q4\6-2b-check-depth-two-ghosts.test

\*\*\* PASS: test\_cases\q4\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 2 seconds.

\*\*\* Won 0 out of 1 games. Average score: 84.000000 \*\*\*

\*\*\* PASS: test\_cases\q4\7-pacman-game.test

### Question q4: 5/5 ###

Question q5

===========

Pacman emerges victorious! Score: 1083

Pacman emerges victorious! Score: 1145

Pacman emerges victorious! Score: 1009

Pacman emerges victorious! Score: 1081

Pacman emerges victorious! Score: 994

Pacman emerges victorious! Score: 1097

Pacman emerges victorious! Score: 1088

Pacman emerges victorious! Score: 1007

Pacman emerges victorious! Score: 1112

Pacman emerges victorious! Score: 1100

Average Score: 1071.6

Scores: 1083.0, 1145.0, 1009.0, 1081.0, 994.0, 1097.0, 1088.0, 1007.0, 1112.0, 1100.0

Win Rate: 10/10 (1.00)

Record: Win, Win, Win, Win, Win, Win, Win, Win, Win, Win

\*\*\* PASS: test\_cases\q5\grade-agent.test (6 of 6 points)

\*\*\* 1071.6 average score (2 of 2 points)

\*\*\* Grading scheme:

\*\*\* < 500: 0 points

\*\*\* >= 500: 1 points

\*\*\* >= 1000: 2 points

\*\*\* 10 games not timed out (1 of 1 points)

\*\*\* Grading scheme:

\*\*\* < 0: fail

\*\*\* >= 0: 0 points

\*\*\* >= 10: 1 points

\*\*\* 10 wins (3 of 3 points)

\*\*\* Grading scheme:

\*\*\* < 1: fail

\*\*\* >= 1: 1 points

\*\*\* >= 5: 2 points

\*\*\* >= 10: 3 points

### Question q5: 6/6 ###

Finished at 15:25:35

**GITLOG:**

commit cb3a323594d2dbd762f829c1930d6cceb71eac7c (HEAD -> master, origin/master)

Author: nishitmehta1 <nishit.mehta1@gmail.com>

Date: Wed Oct 17 13:22:42 2018 -0500

Expectimax Begin

commit c5292780785eb1311585aeea5873f39d383a3af6

Author: nishitmehta1 <nishit.mehta1@gmail.com>

Date: Mon Oct 15 13:55:45 2018 -0500

Alpha-Beta Pruning Update

commit 907105cdbe7f1df9cc4b4fed85df4df6cb9246fd

Author: nishitmehta1 <nishit.mehta1@gmail.com>

Date: Mon Oct 8 18:02:12 2018 -0500

Question 1 and 2 updates

commit 39c395dcddc0c2d4f94312288d9d5a68d492a276

Author: nishitmehta1 <nishit.mehta1@gmail.com>

Date: Sun Oct 7 18:03:50 2018 -0500

First Commit