Show your autograder results and describe each algorithm:

- Q1. Reflex Agent (2%)

```
Question q1
===========

Pacman emerges victorious! Score: 1219
Pacman emerges victorious! Score: 1228
Pacman emerges victorious! Score: 1231
Pacman emerges victorious! Score: 1231
Pacman emerges victorious! Score: 1228
Pacman emerges victorious! Score: 1231
Pacman emerges victorious! Score: 1215
Pacman emerges victorious! Score: 1227
Pacman emerges victorious! Score: 1231
Pacman emerges victorious! Score: 1231
Average Score: 1227.2
Scores:        1219.0, 1228.0, 1231.0, 1231.0, 1228.0, 1231.0, 1215.0, 1227.0, 1231.0, 1231.0
Win Rate:      10/10 (1.00)
Record:        Win, Win, Win, Win, Win, Win, Win, Win, Win, Win
*** PASS: test_cases\q1\grade-agent.test (30.0 of 30.0 points)
***     1227.2 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: 30/30 ###
```

This algorithm focuses on the immediate outcomes. It tries to access the current state of the agent and enemies and food. It analyzes the current state to determine whether move away from the enemies or get closer to the food.

- Q2. Minimax (2%)

```
Question q2
===========

*** PASS: test_cases\q2\0-eval-function-lose-states-1.test
*** PASS: test_cases\q2\0-eval-function-lose-states-2.test
*** PASS: test_cases\q2\0-eval-function-win-states-1.test
*** PASS: test_cases\q2\0-eval-function-win-states-2.test
*** 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 0 seconds.
```

```
Record:        Loss
*** Finished running MinimaxAgent on smallClassic after 0 seconds.
*** Won 0 out of 1 games. Average score: 84.000000 ***
*** PASS: test_cases\q2\8-pacman-game.test

### Question q2: 30/30 ###
```

In every round, this algorithm traces every possible move of the player and the enemies, continuing to trace their moves until the depth equals to 0. For the player,

it selects the best move that maximizes the player's advantage. Conversely, for the enemies, it assumes the worst possible moves from the player's perspective, meaning the enemies aim to reduce the player's score. The final score helps to determine the action the player will take in this round.

● Q3. Alpha-Beta Pruning (2%)

```
Question q3
===========

*** PASS: test_cases\q3\0-eval-function-lose-states-1.test
*** PASS: test_cases\q3\0-eval-function-lose-states-2.test
*** PASS: test_cases\q3\0-eval-function-win-states-1.test
*** PASS: test_cases\q3\0-eval-function-win-states-2.test
*** 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 0 seconds.
*** Won 0 out of 1 games. Average score: 84.000000 ***
*** PASS: test_cases\q3\8-pacman-game.test

### Question q3: 30/30 ###


Finished at 1:11:27

Provisional grades
==================
Question q1: 30/30
Question q2: 30/30
Question q3: 30/30
------------------
Total: 90/90
```

This algorithm is an optimization technique for the Minimax algorithm. It traces every possible step of the player and the enemies. When it is found that a specific subtree performs worse (yields a lower score) than the best score already found, the algorithm stops considering that path because it is deemed not to lead to an optimal solution. In this way, while considering every possible step of each entity, the algorithm prunes redundant solutions, thus making the determination process more efficient.

■ Describe the idea of your design about evaluation function in Q1. (2%)

My design for the evaluation function in Q1 considers both food and the enemy. There are two scenarios: a) very close to the enemy, and b) not that close to the enemy. In the first scenario, we should encourage the player to move away from the enemy while still attempting to eat food, so we return a very low score to reflect the danger. In the second scenario, while the focus is on acquiring food, it's still important to remain cautious of the enemy. Thus, we update the score with both the distance to the food and the distance from the enemy. This way, the evaluation function is constructed to balance safety and goal achievement.

■ Demonstrate the speed up after the implementation of pruning. (2%)

```
user@DESKTOP-FN5A9LV MINGW64 ~/Desktop/AI/AI2024-hw2/AI2024-hw2
$ python pacman.py -p MinimaxAgent -a depth=3 -l smallClassic
Pacman emerges victorious! Score: 1421
Average Score: 1421.0
Scores:        1421.0
Win Rate:      1/1 (1.00)
Record:        Win
```

```
user@DESKTOP-FN5A9LV MINGW64 ~/Desktop/AI/AI2024-hw2/AI2024-hw2
$ python pacman.py -p AlphaBetaAgent -a depth=3 -l smallClassic
Pacman died! Score: 10
Average Score: 10.0
Scores:        10.0
Win Rate:      0/1 (0.00)
Record:        Loss
```

These two photos demonstrate the speed up after the implementation of pruning.