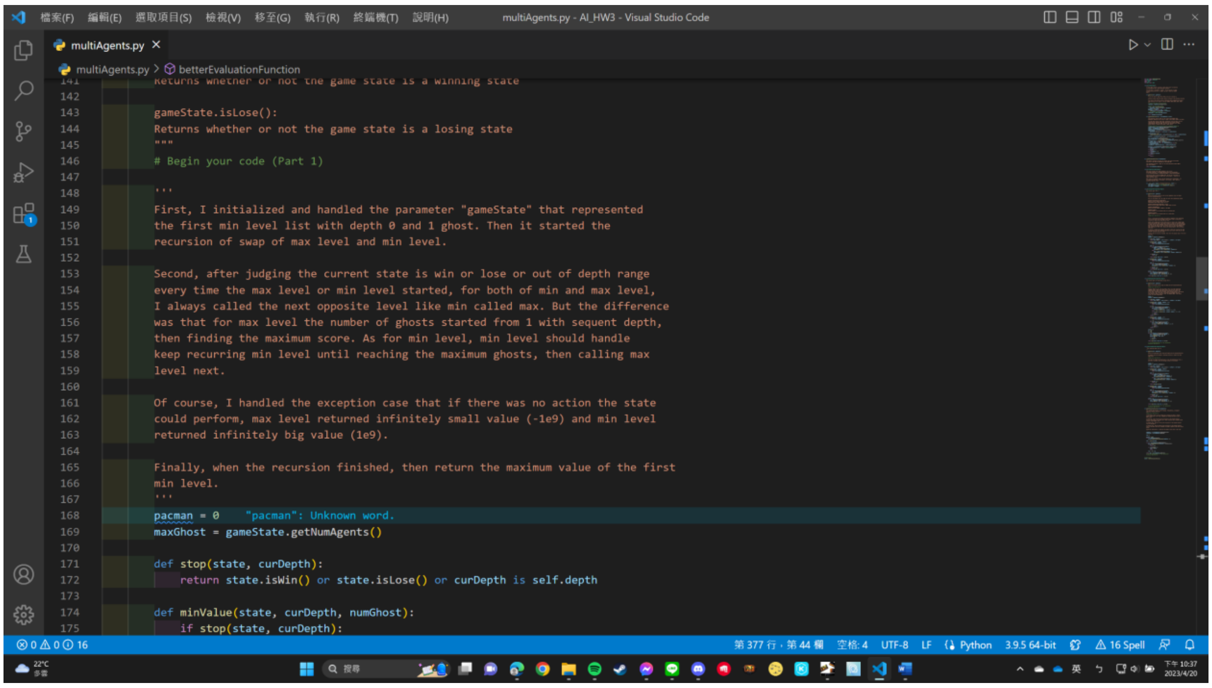
# Homework 3: Multi-Agent Search

Please keep the title of each section and delete examples.

## Part I. Implementation (5%):

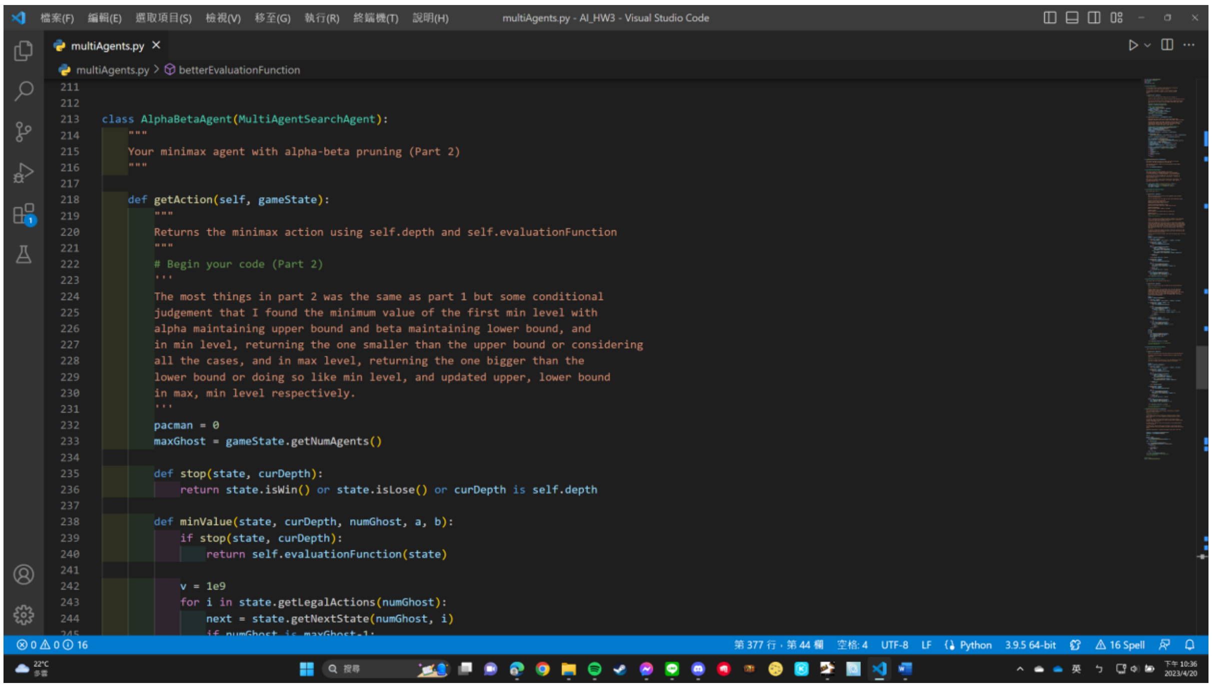
- Please screenshot your code snippets of Part 1 ~ Part 4, and explain your implementation. For example,

Part 1:



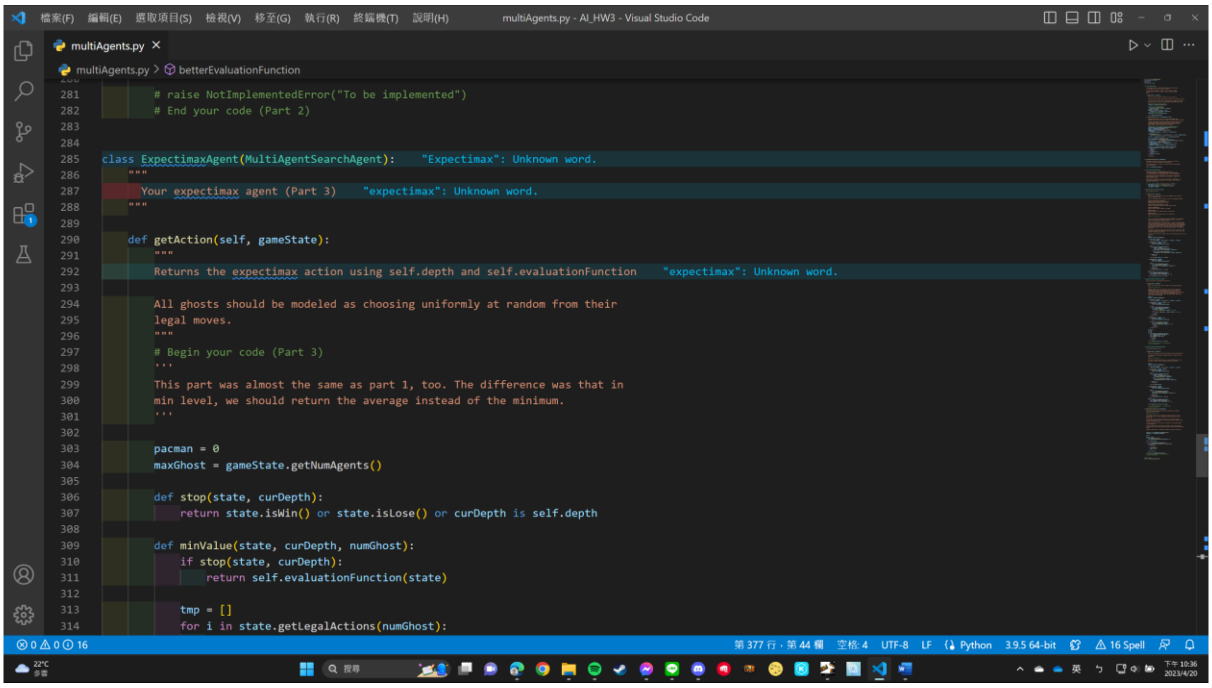
```
    Returns whether or not the game state is a winning state

    gameState.isLose():
    Returns whether or not the game state is a losing state
    """
    # Begin your code (Part 1)

    '''
    First, I initialized and handled the parameter "gameState" that represented
    the first min level list with depth 0 and 1 ghost. Then it started the
    recursion of swap of max level and min level.

    Second, after judging the current state is win or lose or out of depth range
    every time the max level or min level started, for both of min and max level,
    I always called the next opposite level like min called max. But the difference
    was that for max level the number of ghosts started from 1 with sequent depth,
    then finding the maximum score. As for min level, min level should handle
    keep recurring min level until reaching the maximum ghosts, then calling max
    level next.

    Of course, I handled the exception case that if there was no action the state
    could perform, max level returned infinitely small value (-1e9) and min level
    returned infinitely big value (1e9).

    Finally, when the recursion finished, then return the maximum value of the first
    min level.
    '''
    pacman = 0    "pacman": Unknown word.
    maxGhost = gameState.getNumAgents()

    def stop(state, curDepth):
        return state.isWin() or state.isLose() or curDepth is self.depth

    def minValue(state, curDepth, numGhost):
        if stop(state, curDepth):
```

Part 2:


檔案(F) 編輯(E) 選取項目(S) 檢視(V) 移至(G) 執行(R) 終端機(T) 說明(H)
multiAgents.py - AI_HW3 - Visual Studio Code
multiAgents.py
multiAgents.py > betterEvaluationFunction
⊗ 0 ⚠ 0 ⓘ 16
第 377 行，第 44 欄 空格: 4 UTF-8 LF Python 3.9.5 64-bit ⚠ 16 Spell

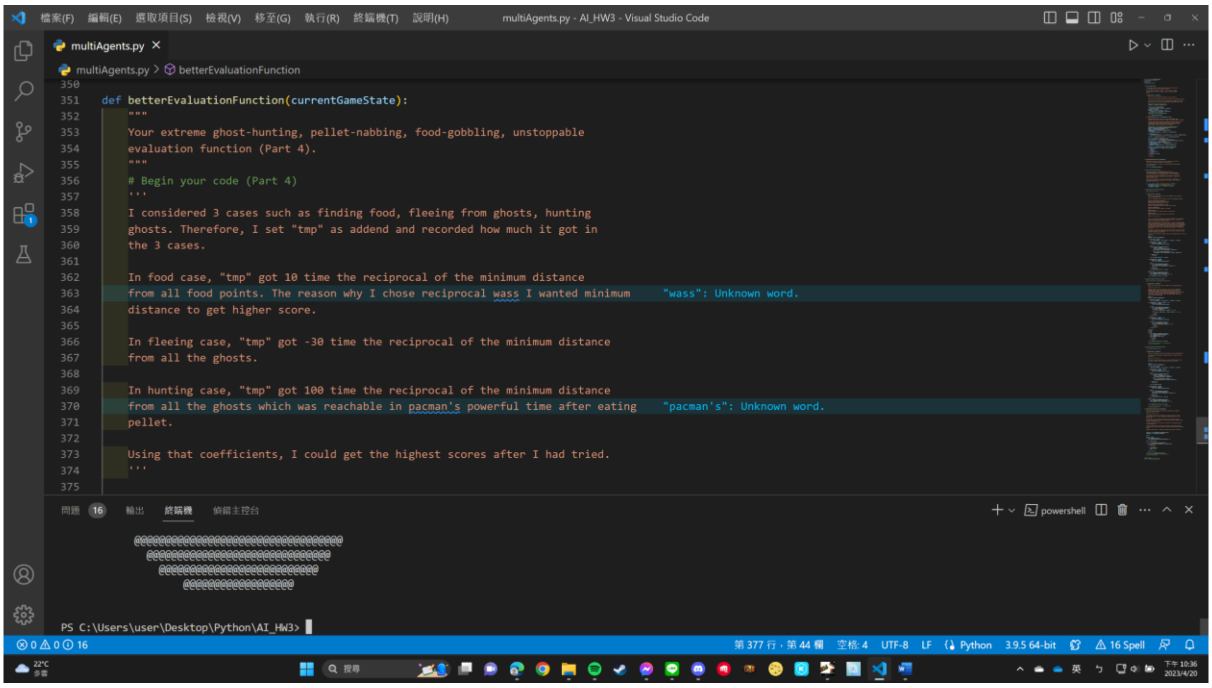
```python


class AlphaBetaAgent(MultiAgentSearchAgent):
    """
    Your minimax agent with alpha-beta pruning (Part 2)
    """

    def getAction(self, gameState):
        """
        Returns the minimax action using self.depth and self.evaluationFunction
        """
        # Begin your code (Part 2)
        '''
        The most things in part 2 was the same as part 1 but some conditional
        judgement that I found the minimum value of the first min level with
        alpha maintaining upper bound and beta maintaining lower bound, and
        in min level, returning the one smaller than the upper bound or considering
        all the cases, and in max level, returning the one bigger than the
        lower bound or doing so like min level, and updated upper, lower bound
        in max, min level respectively.
        '''
        pacman = 0
        maxGhost = gameState.getNumAgents()

        def stop(state, curDepth):
            return state.isWin() or state.isLose() or curDepth is self.depth

        def minValue(state, curDepth, numGhost, a, b):
            if stop(state, curDepth):
                return self.evaluationFunction(state)

            v = 1e9
            for i in state.getLegalActions(numGhost):
                next = state.getNextState(numGhost, i)
                if numGhost is maxGhost-1:
```

Part 3:


檔案(F) 編輯(E) 選取項目(S) 檢視(V) 移至(G) 執行(R) 終端機(T) 說明(H)
multiAgents.py - AI_HW3 - Visual Studio Code
multiAgents.py
multiAgents.py > betterEvaluationFunction
⊗ 0 ⚠ 0 ⓘ 16
第 377 行，第 44 欄 空格: 4 UTF-8 LF Python 3.9.5 64-bit ⚠ 16 Spell

```python
        # raise NotImplementedError("To be implemented")
        # End your code (Part 2)


class ExpectimaxAgent(MultiAgentSearchAgent):    "Expectimax": Unknown word.
    """
    Your expectimax agent (Part 3)    "expectimax": Unknown word.
    """

    def getAction(self, gameState):
        """
        Returns the expectimax action using self.depth and self.evaluationFunction    "expectimax": Unknown word.

        All ghosts should be modeled as choosing uniformly at random from their
        legal moves.
        """
        # Begin your code (Part 3)
        '''
        This part was almost the same as part 1, too. The difference was that in
        min level, we should return the average instead of the minimum.
        '''

        pacman = 0
        maxGhost = gameState.getNumAgents()

        def stop(state, curDepth):
            return state.isWin() or state.isLose() or curDepth is self.depth

        def minValue(state, curDepth, numGhost):
            if stop(state, curDepth):
                return self.evaluationFunction(state)

            tmp = []
            for i in state.getLegalActions(numGhost):
```

Part 4:



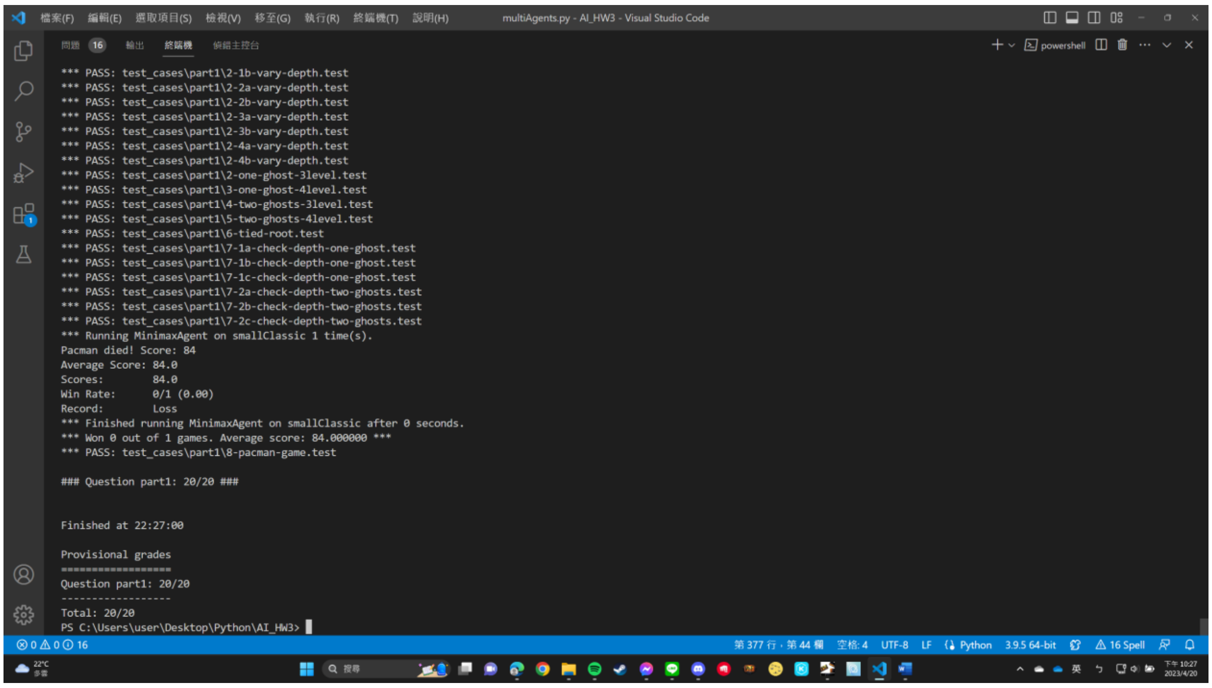
```python
def betterEvaluationFunction(currentGameState):
    """
    Your extreme ghost-hunting, pellet-nabbing, food-gobbling, unstoppable
    evaluation function (Part 4).
    """
    # Begin your code (Part 4)
    '''
    I considered 3 cases such as finding food, fleeing from ghosts, hunting
    ghosts. Therefore, I set "tmp" as addend and recorded how much it got in
    the 3 cases.

    In food case, "tmp" got 10 time the reciprocal of the minimum distance
    from all food points. The reason why I chose reciprocal wass I wanted minimum
    distance to get higher score.

    In fleeing case, "tmp" got -30 time the reciprocal of the minimum distance
    from all the ghosts.

    In hunting case, "tmp" got 100 time the reciprocal of the minimum distance
    from all the ghosts which was reachable in pacman's powerful time after eating
    pellet.

    Using that coefficients, I could get the highest scores after I had tried.
    '''
```

```
PS C:\Users\user\Desktop\Python\AI_HW3>
```

## Part II. Results & Analysis (5%):

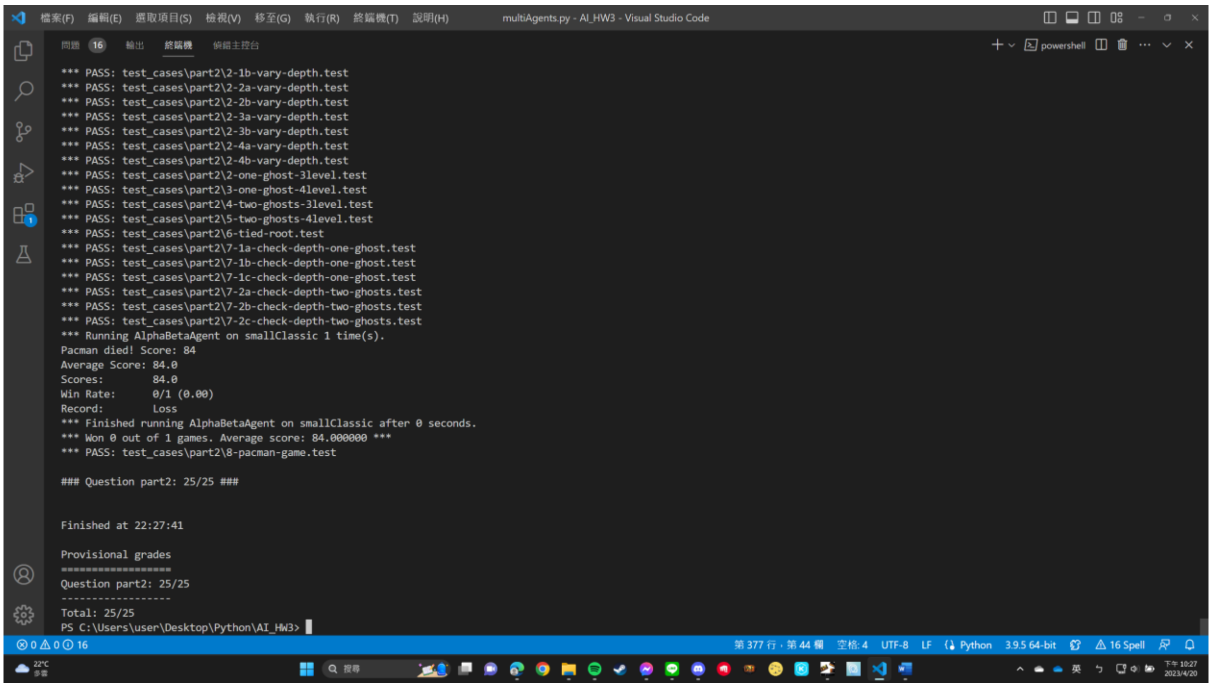
- Please screenshot the results. For instance, the result of the **autograder** and any observation of your **evaluation function**.

Part 1:



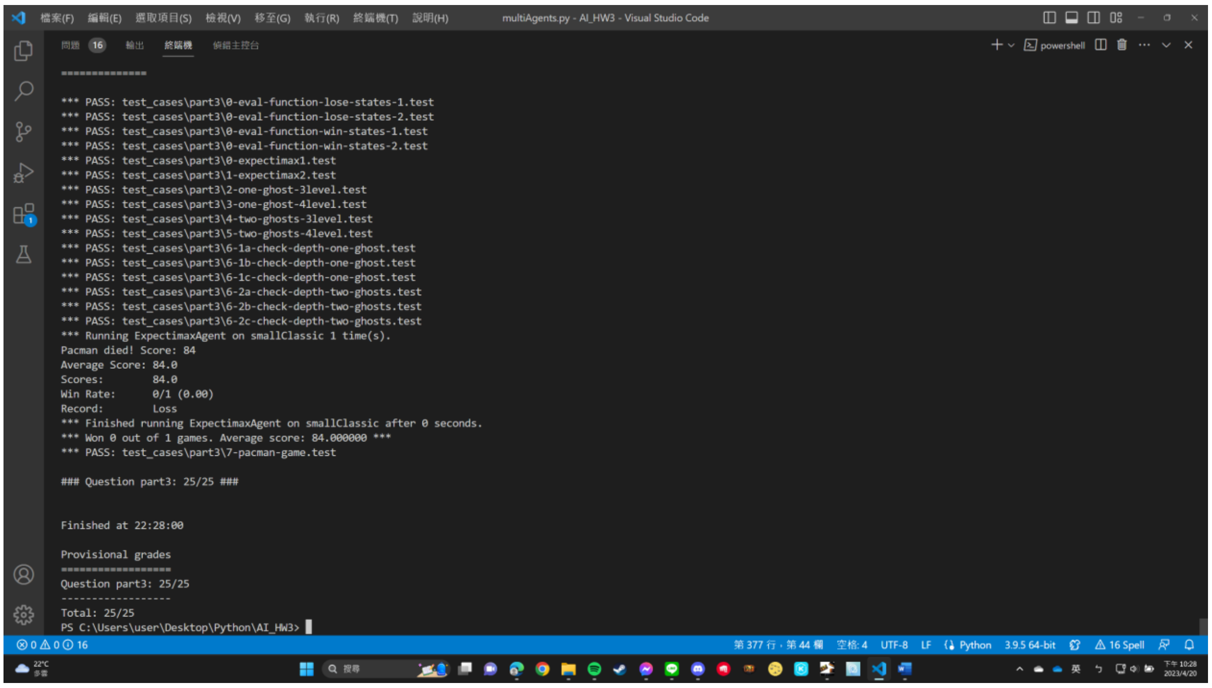
```
*** PASS: test_cases\part1\2-1b-vary-depth.test
*** PASS: test_cases\part1\2-2a-vary-depth.test
*** PASS: test_cases\part1\2-2b-vary-depth.test
*** PASS: test_cases\part1\2-3a-vary-depth.test
*** PASS: test_cases\part1\2-3b-vary-depth.test
*** PASS: test_cases\part1\2-4a-vary-depth.test
*** PASS: 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 0 seconds.
*** Won 0 out of 1 games. Average score: 84.000000 ***
*** PASS: test_cases\part1\8-pacman-game.test

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


Finished at 22:27:00

Provisional grades
==================
Question part1: 20/20
------------------
Total: 20/20
PS C:\Users\user\Desktop\Python\AI_HW3>
```

Part 2:



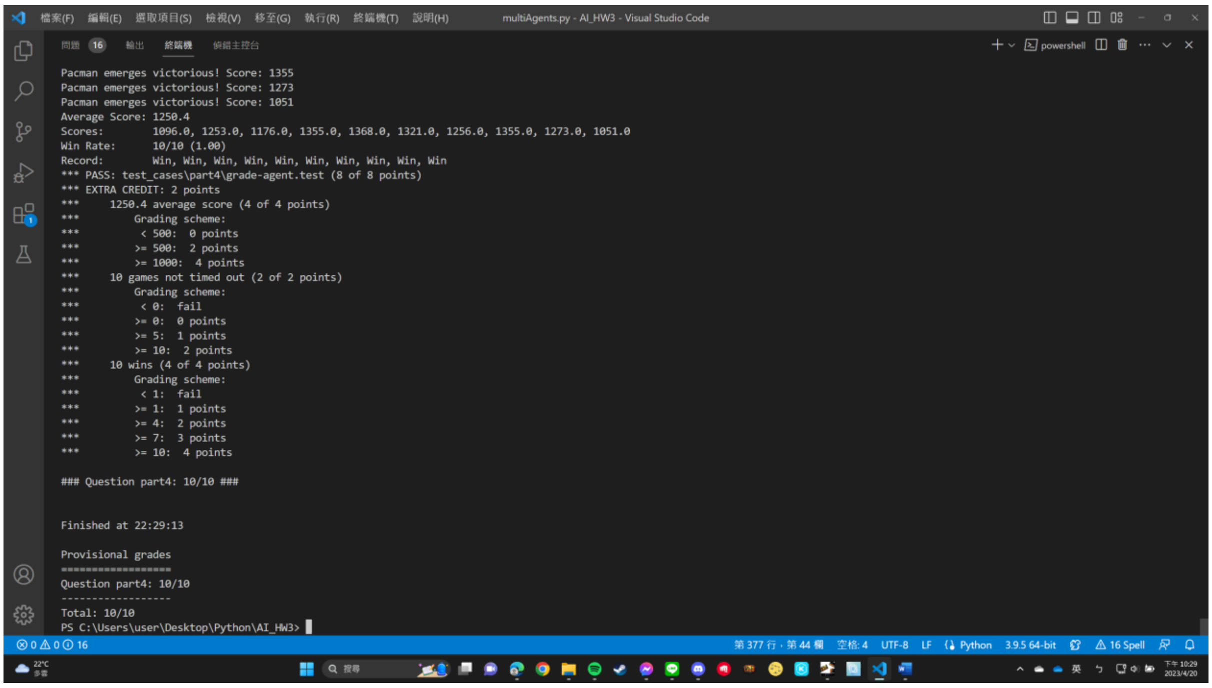
```
*** PASS: test_cases\part2\2-1b-vary-depth.test
*** PASS: test_cases\part2\2-2a-vary-depth.test
*** PASS: test_cases\part2\2-2b-vary-depth.test
*** PASS: test_cases\part2\2-3a-vary-depth.test
*** PASS: test_cases\part2\2-3b-vary-depth.test
*** PASS: test_cases\part2\2-4a-vary-depth.test
*** PASS: 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 ###


Finished at 22:27:41

Provisional grades
==================
Question part2: 25/25
------------------
Total: 25/25
PS C:\Users\user\Desktop\Python\AI_HW3>
```

Part 3:



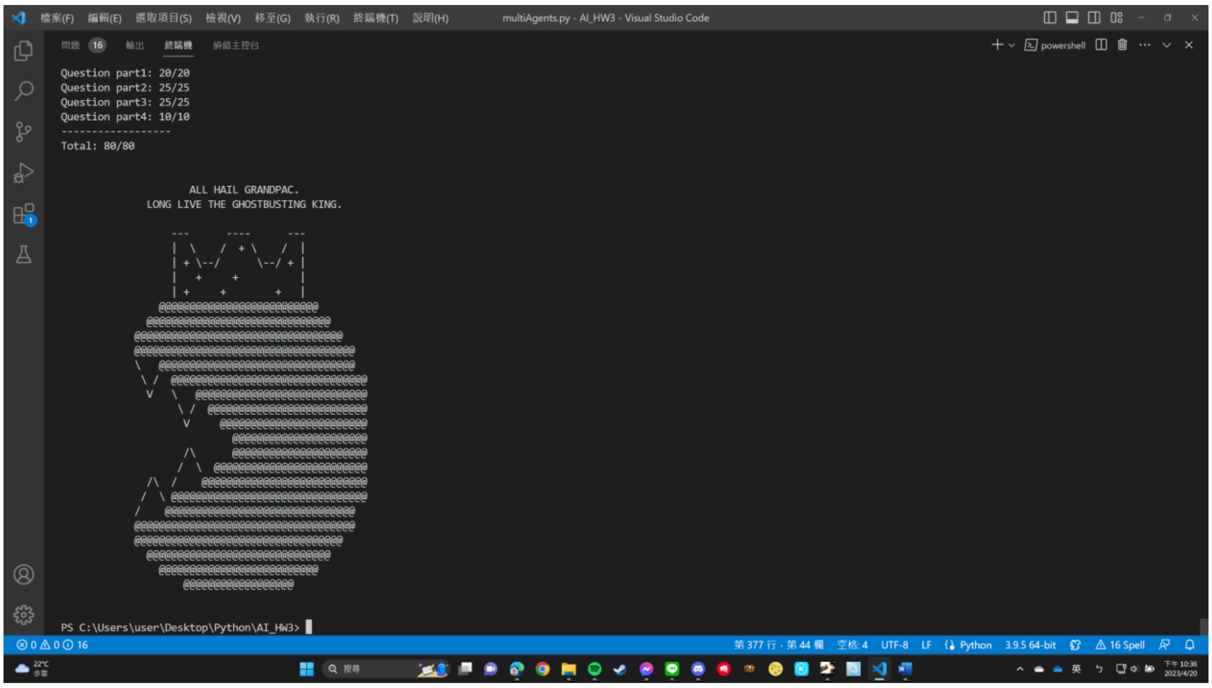
```
==============

*** 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 ###


Finished at 22:28:00

Provisional grades
==================
Question part3: 25/25
------------------
Total: 25/25
PS C:\Users\user\Desktop\Python\AI_HW3>
```

Part 4:



```
Pacman emerges victorious! Score: 1355
Pacman emerges victorious! Score: 1273
Pacman emerges victorious! Score: 1051
Average Score: 1250.4
Scores:        1096.0, 1253.0, 1176.0, 1355.0, 1368.0, 1321.0, 1256.0, 1355.0, 1273.0, 1051.0
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
***     1250.4 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 ###


Finished at 22:29:13

Provisional grades
==================
Question part4: 10/10
------------------
Total: 10/10
PS C:\Users\user\Desktop\Python\AI_HW3>
```

In Part 4, I had tried plenty of coefficients. The best was 10 for eatting food, -30 for fleeing, 100 for huntin g ghosts. If the coefficients were 10 times as big as those, python would work very slowly and pacman wou ld die, which I thought it was too big and overflow. If the coefficients were 10 times as small as those, sinc e I always added a very little amount to score when pacman finding food, thus that would be small enough to be affected. And I thought eatting ghost was important the most to avoid death, fleeing from ghost was t he second, and the last was finding food.



```
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     @@@@@@@@@@@@@@@@@@@@@@@@
                            @@@@@@@@@@@@@@@@@@@@@@
                    /\      @@@@@@@@@@@@@@@@@@@@@@
                   /  \   @@@@@@@@@@@@@@@@@@@@@@@@
              /\  /    @@@@@@@@@@@@@@@@@@@@@@@@@@@
             /  \ @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
            /    @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
            @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
            @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
              @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                @@@@@@@@@@@@@@@@@@@@@@@@@@
                    @@@@@@@@@@@@@@@@@@

PS C:\Users\user\Desktop\Python\AI_HW3>
```