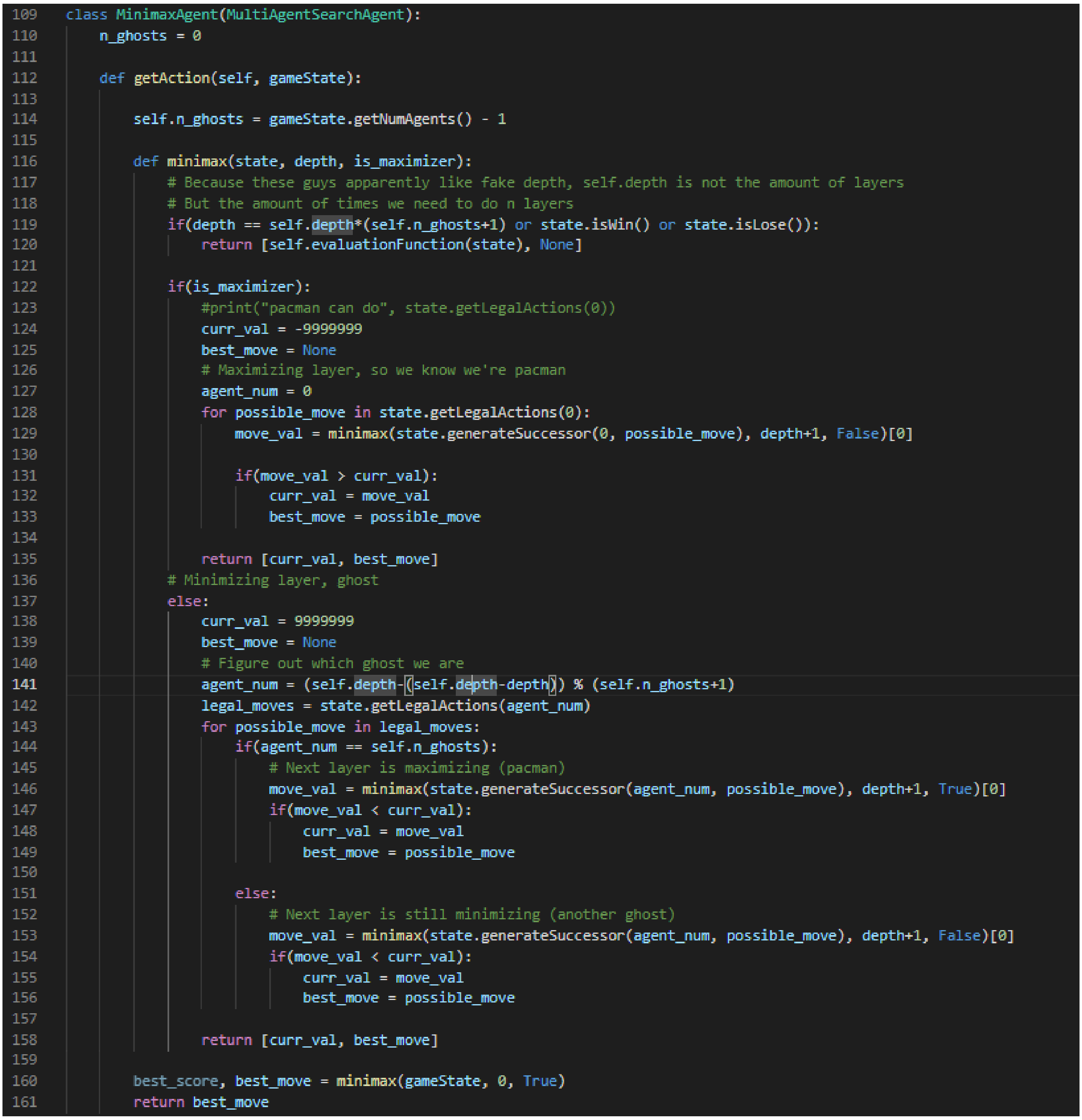
Assignment 3

Minimax and Alpha-beta pruning

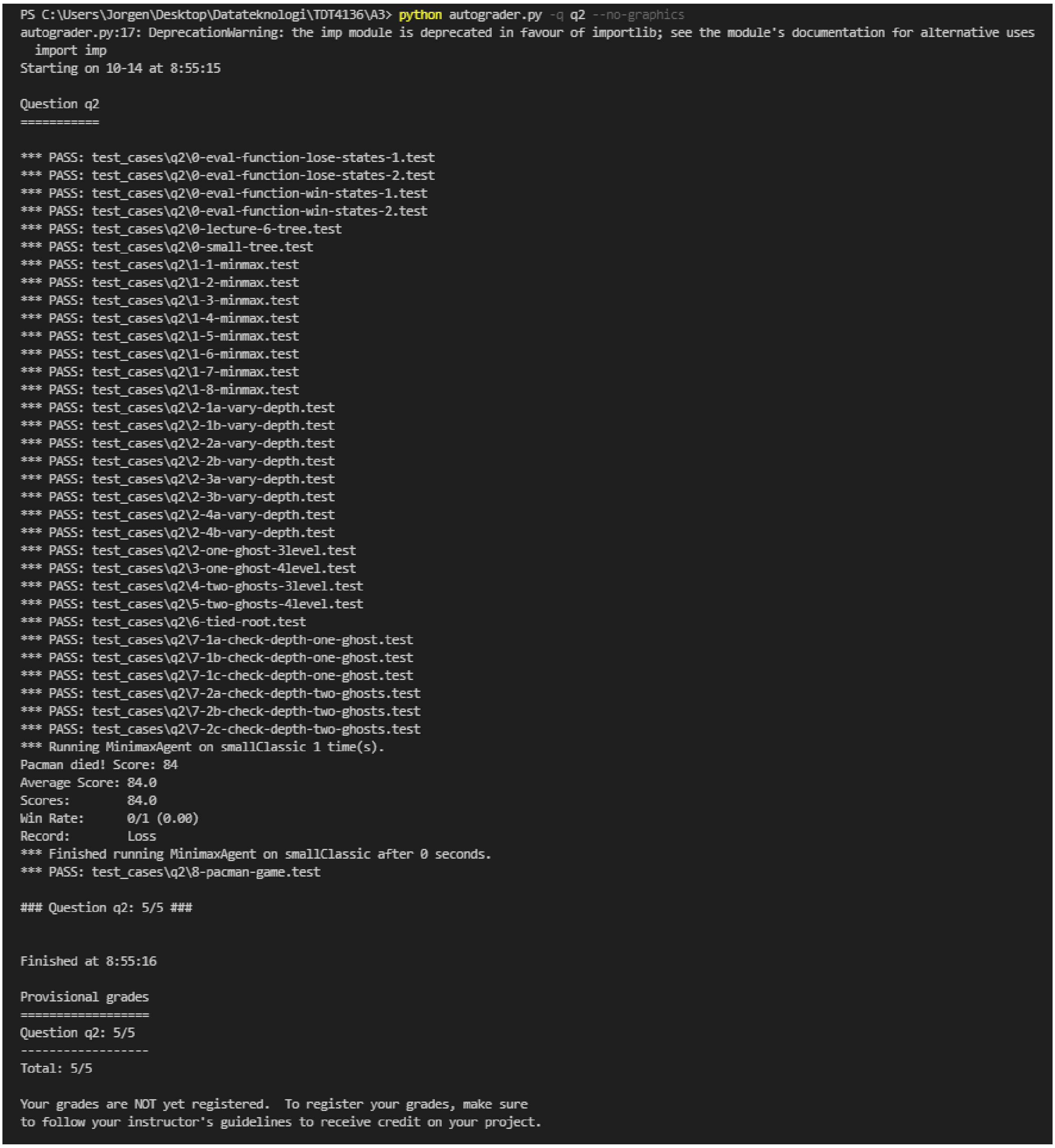
NTNU

Minimax code



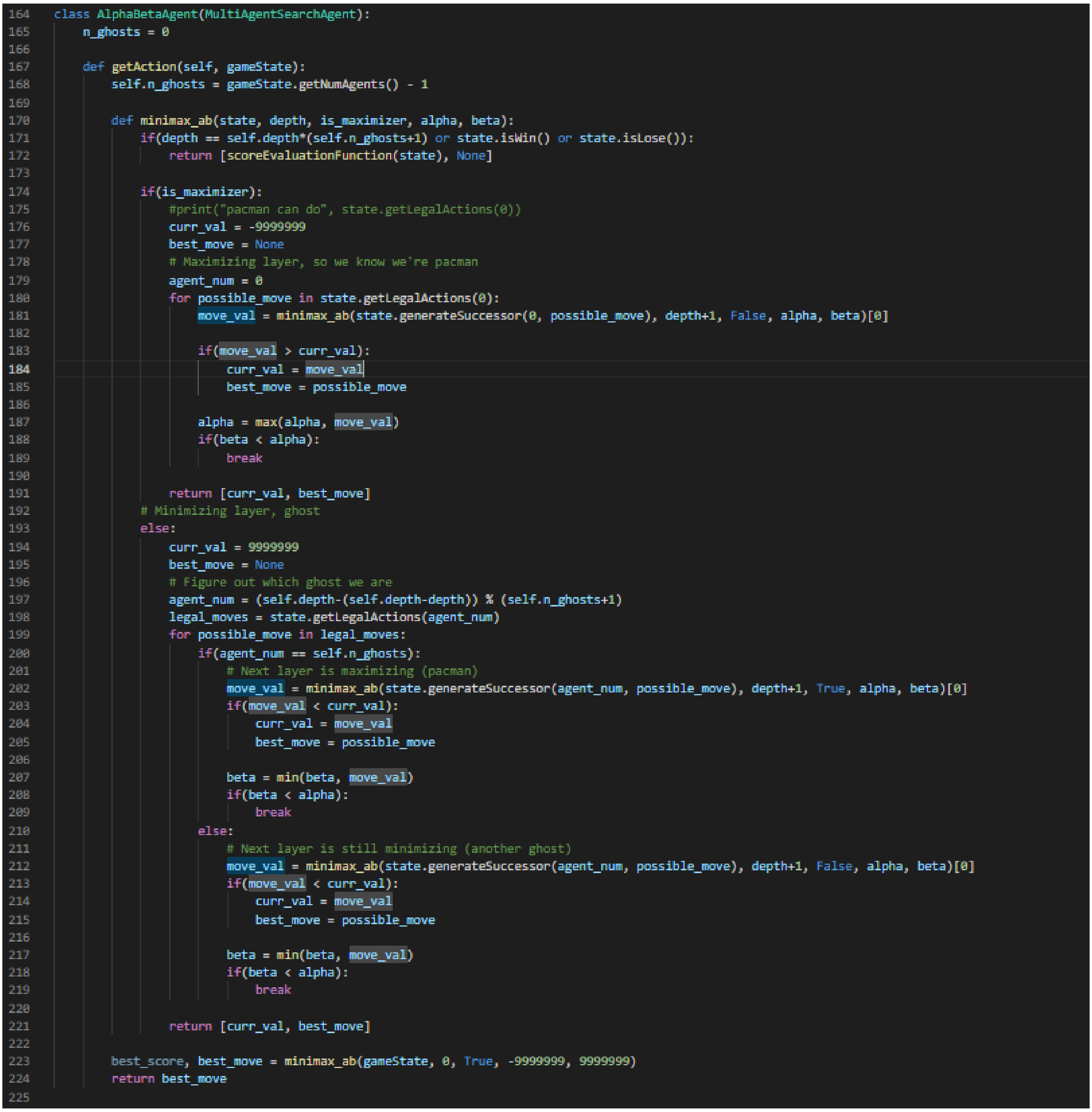
```python
class MinimaxAgent(MultiAgentSearchAgent):
    n_ghosts = 0

    def getAction(self, gameState):

        self.n_ghosts = gameState.getNumAgents() - 1

        def minimax(state, depth, is_maximizer):
            # Because these guys apparently like fake depth, self.depth is not the amount of layers
            # But the amount of times we need to do n layers
            if(depth == self.depth*(self.n_ghosts+1) or state.isWin() or state.isLose()):
                return [self.evaluationFunction(state), None]

            if(is_maximizer):
                #print("pacman can do", state.getLegalActions(0))
                curr_val = -9999999
                best_move = None
                # Maximizing layer, so we know we're pacman
                agent_num = 0
                for possible_move in state.getLegalActions(0):
                    move_val = minimax(state.generateSuccessor(0, possible_move), depth+1, False)[0]

                    if(move_val > curr_val):
                        curr_val = move_val
                        best_move = possible_move

                return [curr_val, best_move]
            # Minimizing layer, ghost
            else:
                curr_val = 9999999
                best_move = None
                # Figure out which ghost we are
                agent_num = (self.depth-(self.depth-depth)) % (self.n_ghosts+1)
                legal_moves = state.getLegalActions(agent_num)
                for possible_move in legal_moves:
                    if(agent_num == self.n_ghosts):
                        # Next layer is maximizing (pacman)
                        move_val = minimax(state.generateSuccessor(agent_num, possible_move), depth+1, True)[0]
                        if(move_val < curr_val):
                            curr_val = move_val
                            best_move = possible_move

                    else:
                        # Next layer is still minimizing (another ghost)
                        move_val = minimax(state.generateSuccessor(agent_num, possible_move), depth+1, False)[0]
                        if(move_val < curr_val):
                            curr_val = move_val
                            best_move = possible_move

                return [curr_val, best_move]

        best_score, best_move = minimax(gameState, 0, True)
        return best_move
```

Minimax output



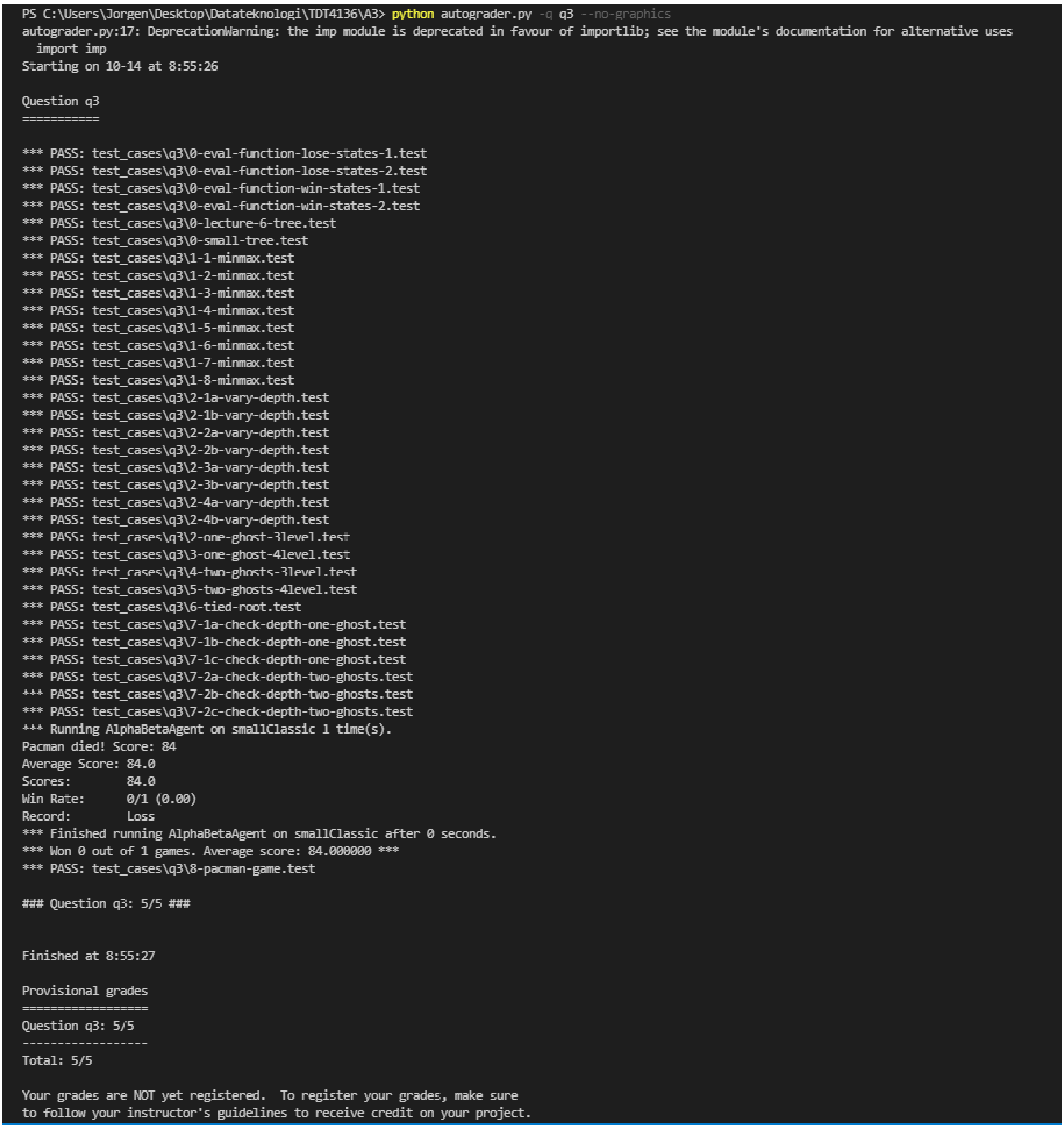
```
PS C:\Users\Jorgen\Desktop\Datateknologi\TDT4136\A3> python autograder.py -q q2 --no-graphics
autograder.py:17: DeprecationWarning: the imp module is deprecated in favour of importlib; see the module's documentation for alternative uses
  import imp
Starting on 10-14 at 8:55:15

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

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


Finished at 8:55:16

Provisional grades
==================
Question q2: 5/5
------------------
Total: 5/5

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.
```

With alpha-beta pruning code



```python
class AlphaBetaAgent(MultiAgentSearchAgent):
    n_ghosts = 0

    def getAction(self, gameState):
        self.n_ghosts = gameState.getNumAgents() - 1

        def minimax_ab(state, depth, is_maximizer, alpha, beta):
            if(depth == self.depth*(self.n_ghosts+1) or state.isWin() or state.isLose()):
                return [scoreEvaluationFunction(state), None]

            if(is_maximizer):
                #print("pacman can do", state.getLegalActions(0))
                curr_val = -9999999
                best_move = None
                # Maximizing layer, so we know we're pacman
                agent_num = 0
                for possible_move in state.getLegalActions(0):
                    move_val = minimax_ab(state.generateSuccessor(0, possible_move), depth+1, False, alpha, beta)[0]

                    if(move_val > curr_val):
                        curr_val = move_val
                        best_move = possible_move

                    alpha = max(alpha, move_val)
                    if(beta < alpha):
                        break

                return [curr_val, best_move]
            # Minimizing layer, ghost
            else:
                curr_val = 9999999
                best_move = None
                # Figure out which ghost we are
                agent_num = (self.depth-(self.depth-depth)) % (self.n_ghosts+1)
                legal_moves = state.getLegalActions(agent_num)
                for possible_move in legal_moves:
                    if(agent_num == self.n_ghosts):
                        # Next layer is maximizing (pacman)
                        move_val = minimax_ab(state.generateSuccessor(agent_num, possible_move), depth+1, True, alpha, beta)[0]
                        if(move_val < curr_val):
                            curr_val = move_val
                            best_move = possible_move

                        beta = min(beta, move_val)
                        if(beta < alpha):
                            break
                    else:
                        # Next layer is still minimizing (another ghost)
                        move_val = minimax_ab(state.generateSuccessor(agent_num, possible_move), depth+1, False, alpha, beta)[0]
                        if(move_val < curr_val):
                            curr_val = move_val
                            best_move = possible_move

                        beta = min(beta, move_val)
                        if(beta < alpha):
                            break

                return [curr_val, best_move]

        best_score, best_move = minimax_ab(gameState, 0, True, -9999999, 9999999)
        return best_move

```

Alpha-beta pruning output



```
PS C:\Users\Jorgen\Desktop\Datateknologi\TDT4136\A3> python autograder.py -q q3 --no-graphics
autograder.py:17: DeprecationWarning: the imp module is deprecated in favour of importlib; see the module's documentation for alternative uses
  import imp
Starting on 10-14 at 8:55:26

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: 5/5 ###


Finished at 8:55:27

Provisional grades
==================
Question q3: 5/5
------------------
Total: 5/5

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.
```