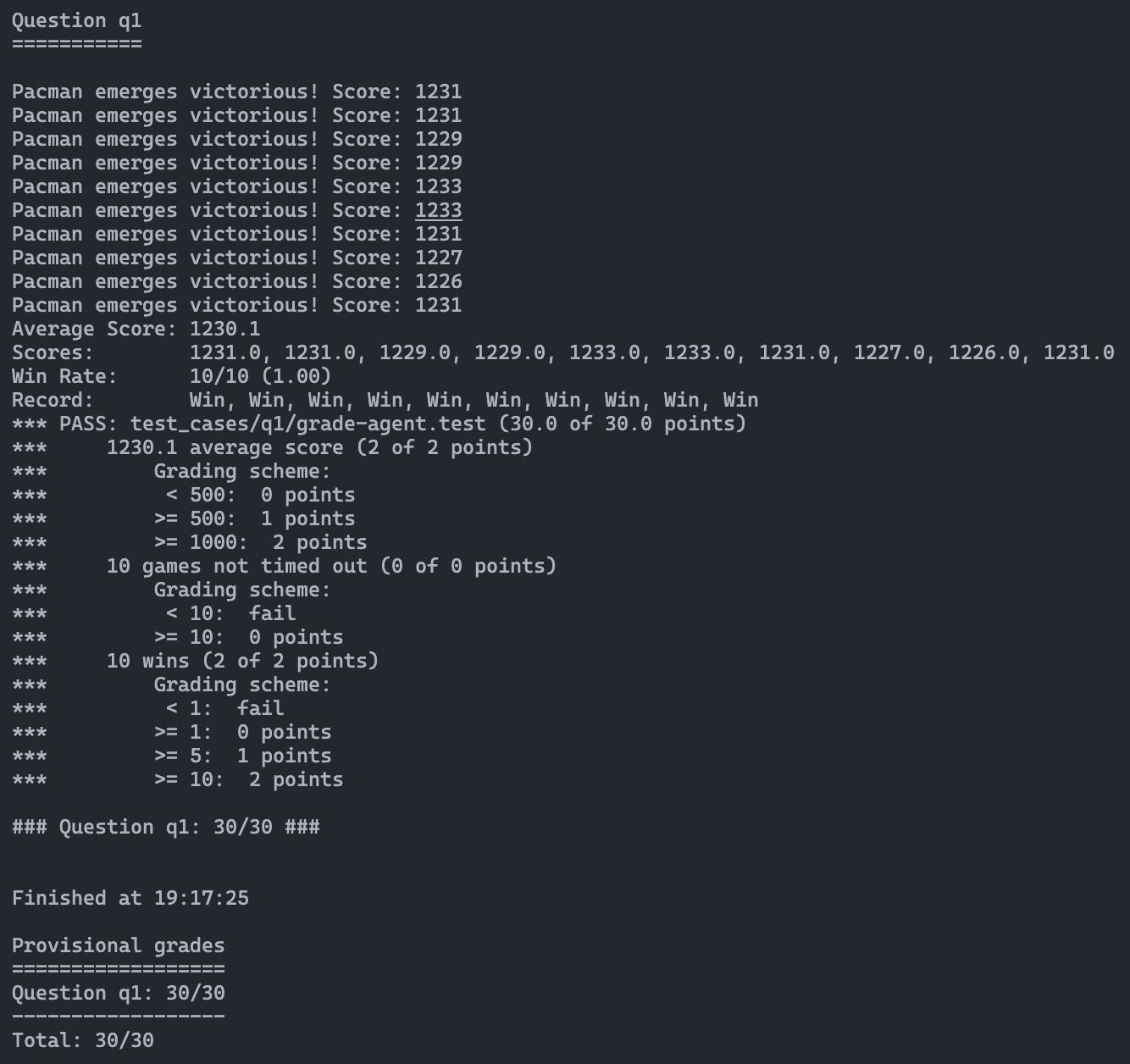
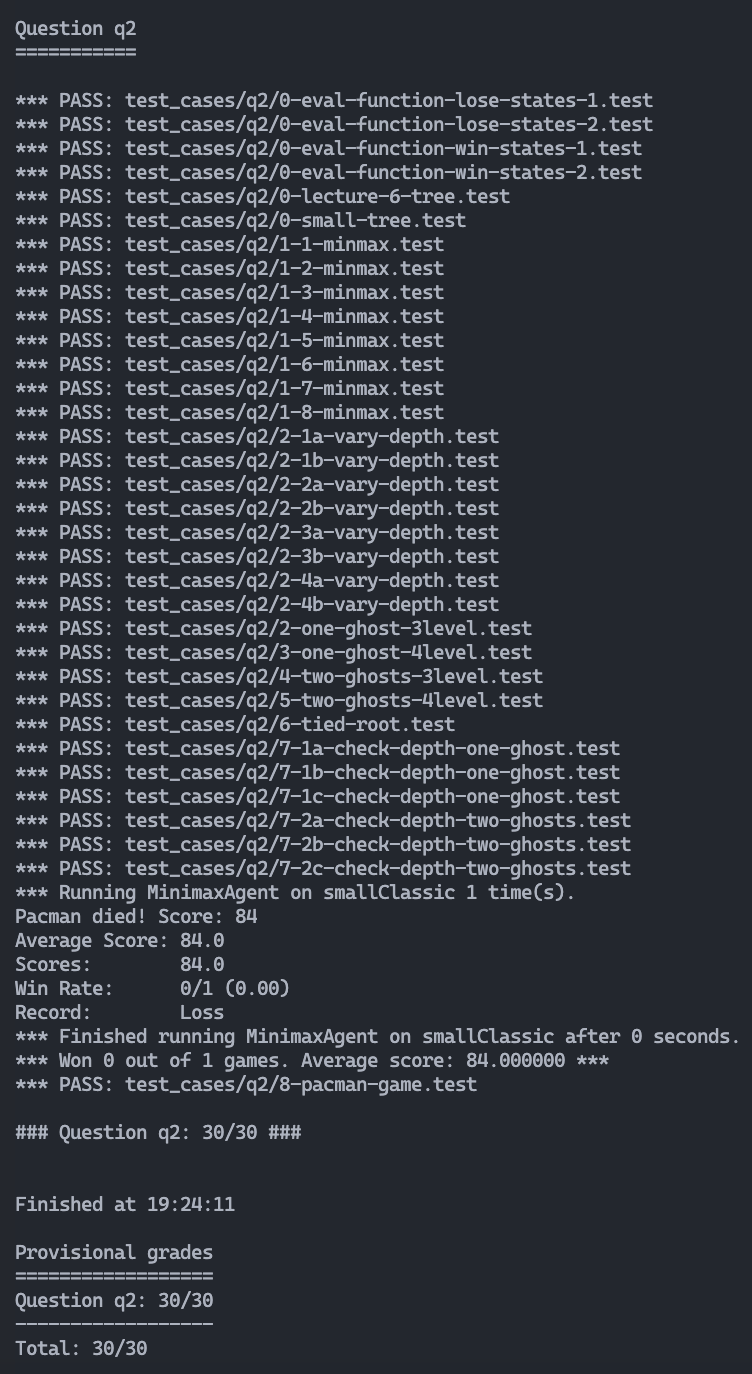
Hw2\_r10723057

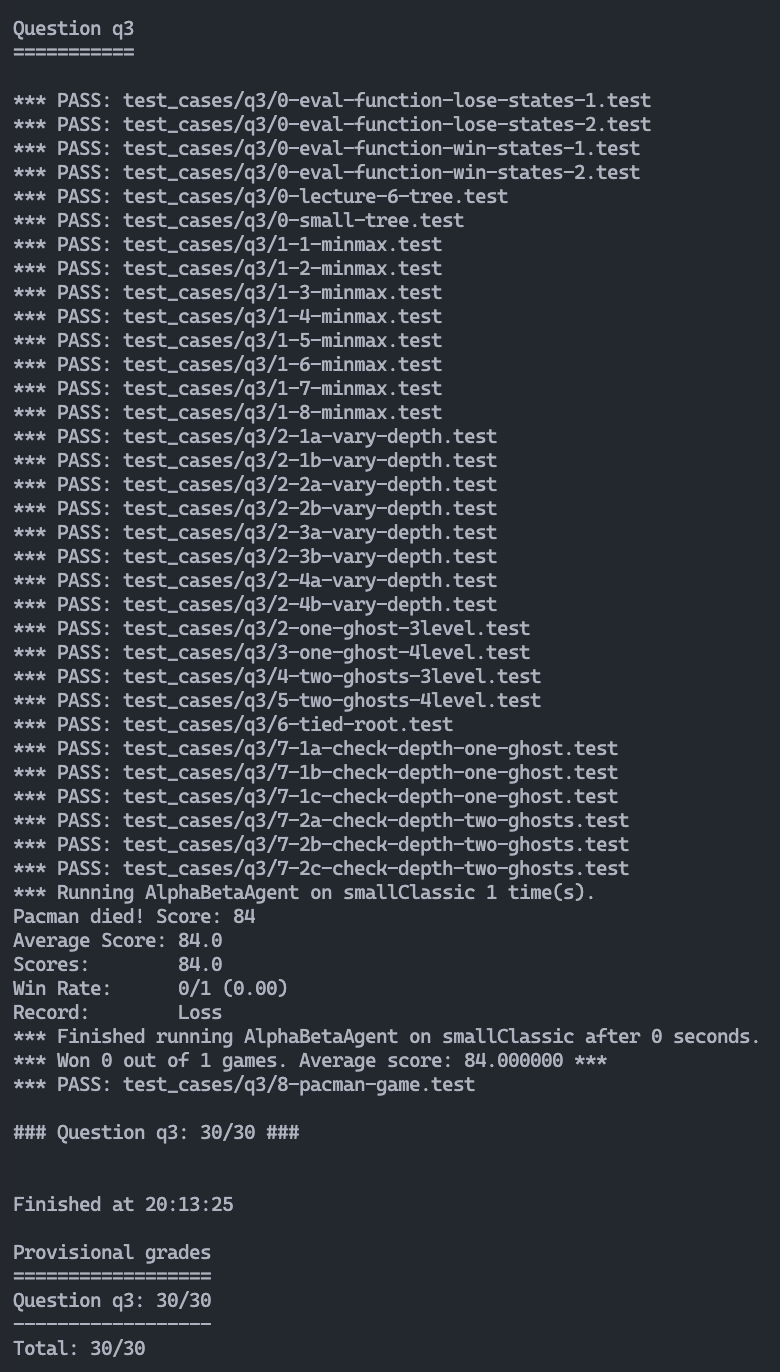
* **Show your autograder results and describe each algorithm:**
* Q1. Reflex Agent

首先在最基本的Reflex Agent，演算法僅僅考慮了與食物之間的距離、考慮鬼魂的位置與驚嚇狀態，並且根據食物的遠近給予分數上的權重、根據和鬼魂的距離來給予分數上的懲罰，最後結合計算的評分和遊戲的基本分數來進行決策。

* Q2. Minimax

首先Pacman的策略是要極大化自己的分數效益；而相反地Ghost的策略是要極小化Pacman的效益。根據投影片上提供的pseudo-code來進行修改，透過遞迴關係來看Pacman的下一步應該要如何行動。修改完max-value, min-value後，加上一個啟動步驟，首先將最佳行動設為None, 最佳分數設為負無限大，然後開始進行演算法，從Pacman開始，透過生出successor，也就是生成執行該行動後的新遊戲狀態，並假設所有Ghost都會選擇對他們最有利的行動時，Pacman能達到的最低評分，若此評分大於先前的最佳評分則更新分數與行動。

* Q3. Alpha-Beta Pruning



和minimax類似，只是利用Alpha-Beta剪枝，來跳過那些不會影響決策的分支，得以節省大幅運算時間。Alpha剪枝：如果當前節點的值v超過beta，則剪枝並返回v。同時更新alpha；Beta剪枝：如果當前節點的值v低於alpha，則剪枝並返回v。同時更新beta。其餘和minimax大致一樣。

* Describe the idea of your design about evaluation function in Q1.

初始想法就是：首先給距離越近的食物越大的權重、距離越遠的食物越小的權重。接著就是設定一個安全距離並考慮鬼魂的位置與驚嚇狀態：離鬼魂接近到一個安全距離內，首先判斷此時鬼魂是否在驚嚇狀態內，若是且離驚嚇狀態結束還有至少兩秒，則我們要接近他必且吃掉；若已經要結束驚嚇狀態或甚至根本不在驚嚇狀態內，則要設定penalty來懲罰近距離接觸，最後將此分數加上原有的評分，即為我們的evaluation function所回傳的分數。

* Demonstrate the speed up after the implementation of pruning.

左邊為Q2(minimax)，右邊為Q3(alpha-beta pruning)，可以看出透過觀察擴展出的states數量，pruning後的states數量每一個確實都小於等於minimax擴展的state.

