**Homework 3: Multi-Agent Search**

**Part I. Implementation (5%):**

**Part 1: Minimax Search**

****

**Part 2: Alpha-Beta Pruning **

**Part 3: Expectimax Search**

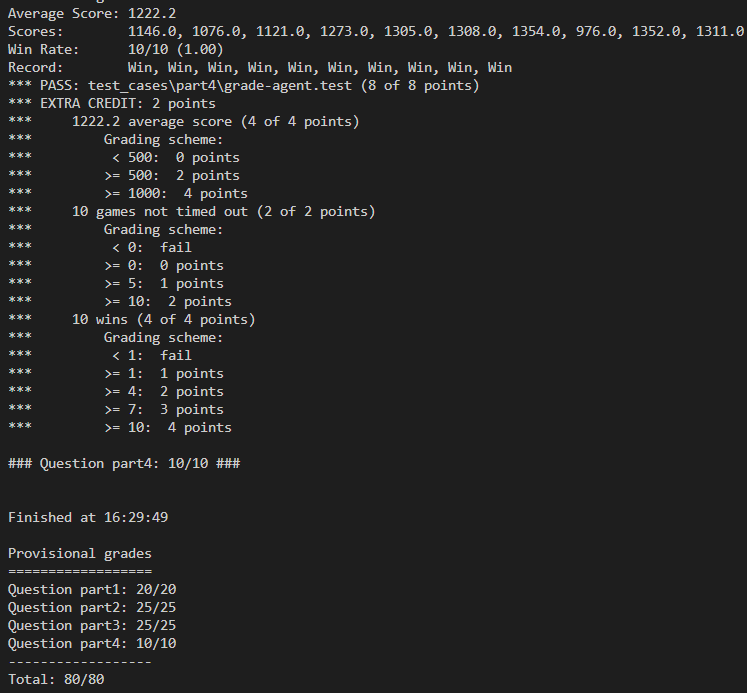
****

**Part 4: Evaluation Function**

****

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

* Result of autograder:



* Observation of my evaluation function: 一張含有 文字 的圖片

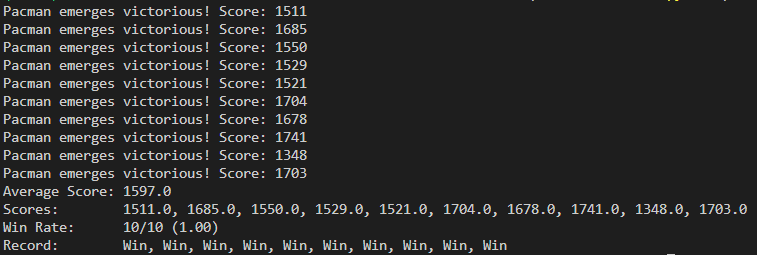
  自動產生的描述 一張含有 文字 的圖片

  自動產生的描述

The game feature and weight are defined in the image above. Negative weight values indicate undesired states, while positive weight values indicate desired states. Based on this score calculation, my strategy is to prioritize killing scared ghosts. To achieve this, I assign a high weight to the "distance to the closest scared ghost" feature and the flag indicating the presence of a nearby scared ghost.  
Although my implementation can pass all the cases in autograder.py, there are still some limitations to it. For example, if you run the code like “python pacman.py -l smallClassic -p ExpectimaxAgent -a evalFn=better -n 10 -q”, the result may not be optimal, as shown in the following image:

一張含有 文字 的圖片

自動產生的描述

However, it may sometimes achieve a decent score, as demonstrated in the following image:

You can try to adjust the weight in the array to see whether you can get a better result.