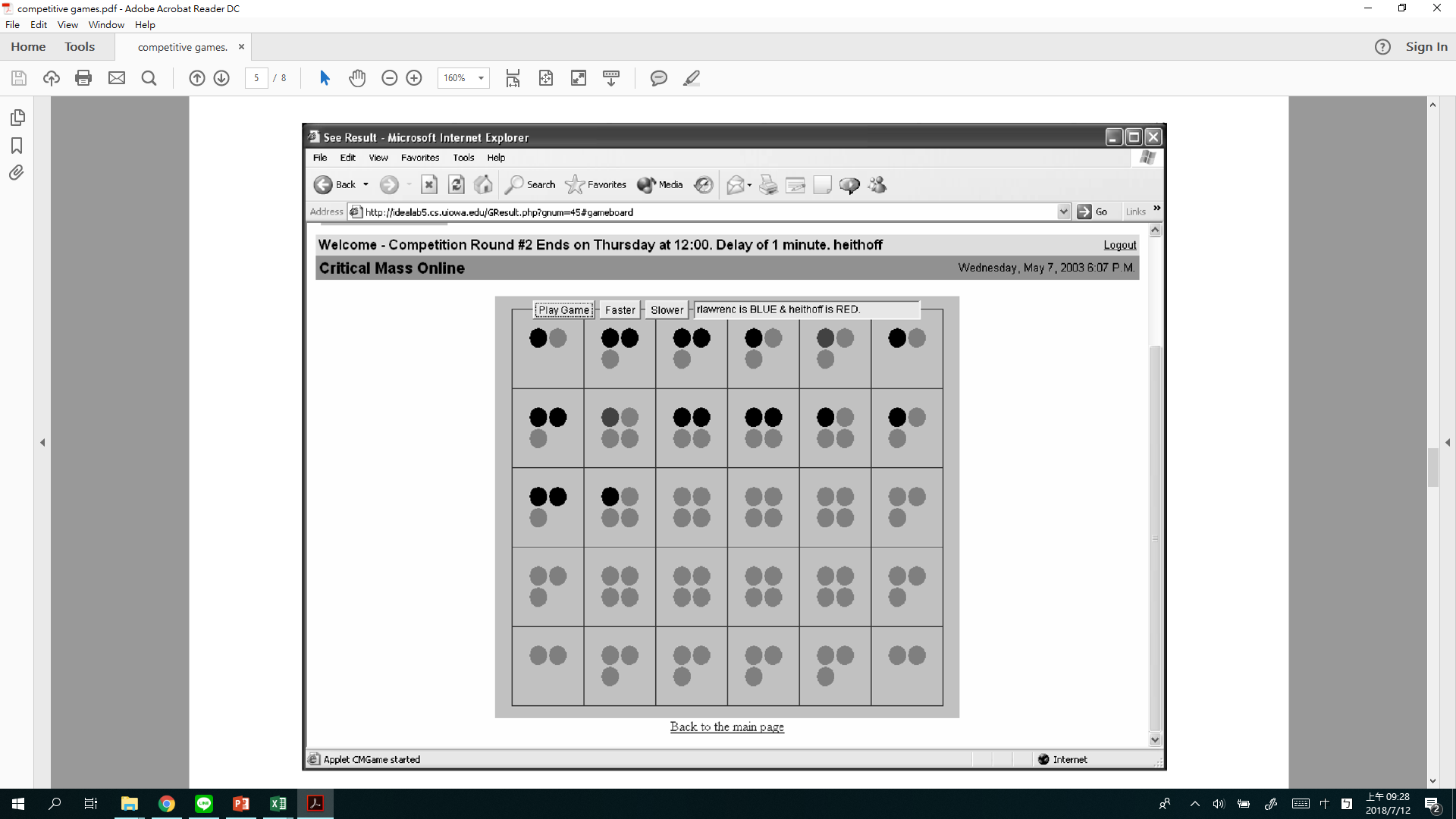
**Project #3: Chain Reaction 連環爆**

1. Project Objective:   
   Apply the knowledge learned from the course of data structures and implement a competitive algorithm for the Critical Mass game.
2. Project Description   
   The gameplay takes place on a 5x6 board of squares. For each cell on the board, we define a critical mass. The critical mass is equal to the number of orthogonally and diagonally adjacent cells. Therefore, that would be 8 for normal cells, 5 for cells in the edge, and 3 for cells in the corner. All cells are initially empty. The Red and the Blue player take turns to place "orbs" of their corresponding colors. Each player can only place an orb of the chosen color either in an empty cell or a cell that already contains the same color orbs. When two or more orbs are placed in the same cell, they stack up. When a cell is loaded with a number of orbs equal to its critical mass, the stack immediately explodes. As a result of the explosion, the cell of explosion loses all orbs and becomes empty. At the same time, an orb is added to each orthogonally and diagonally adjacent cell. All orbs of these orthogonally and diagonally adjacent cells change color to the color of the orbs of explosion. The explosions might result in overloading an adjacent cell, and the chain reaction of the explosion continues until every cell is stable. The winner is the one who eliminates every other player's orbs.



You are to design a competitive algorithm, and TAs will integrate your code to act as one of the game players. Your code is evaluated based on whether their code plays the game correctly and how well it plays against predefined bots.

1. Project Submission Rules
   1. Please use GitHub for source code control and show your program development history. Please follow the version control rules when doing programming.
   2. Grading Policy   
      The basic evaluation is as follows:
      * 15%—if code can be successfully compiled on the competition server and can successfully play one game (regardless of outcome);
      * 30%—if code could beat *randomMove*, a bot that performed random moves;
      * 45%—if code could beat *noLook*, a bot with a board evaluator but no game tree;
      * 60%—if code could beat *heithoff*, a bot with a board evaluator and game tree with four levels of look-ahead;
      * 75%—if code could beat *rlawrenc*, a bot with a very strong board evaluator and game tree with four levels of look-ahead;
      * The winner is the one who first collects 4 wins. For the competition with TA, you will be the starting player.
      * Bonus competition credit will be awarded to every participating student according to the following formula:

where *W* is win rate. TAs will take your algorithm to compete with all other players after the deadline, and the percentage of wins is the win rate *W*. You may continue to improve your algorithm before the deadline.

Two players would take turns to be the starting player. Whoever first collects 4 wins is the winner.

* 1. TA's testing platform
     + OS: Fresh Ubuntu 16.04.
     + Put all codes in one folder.
     + Compiling command:

$g++ chain\_reaction.cpp board.cpp rules.cpp player.cpp algorithm\_ST.cpp algorithm\_TA.cpp

* + - The rules and board setting are written in "rules.cpp" and "board.cpp." Please read the code in those two files before coding.
    - Please name your code as *algorithm\_ST.cpp*, and do not edit other .cpp files provided. In *algorithm\_ST.cpp,* don’t change function name and parameter.



* + - The main data flow is in the chain\_reaction.cpp.
    - The time for placing one orb is limited to 1 second.
    - Please delete all dynamic memory allocation in your algorithm.
    - In *algorithm\_TA.cpp*, a "*randomMove*" code is provided to play as an *opponent* to help you develop your program. You should not hand in your *algorithm\_ST.cpp* code copied from *algorithm\_TA*.cpp.
  1. Change the “Chain Reaction Framework” folder name to "***StudentID*\_proj3**". Create a single compressed zip file from "***StudentID*\_proj3**" to "***StudentID*\_proj3.zip**" for each submission and submit it to the **eLearn** platform. For example, your zip file's name should be "108000299\_proj3.zip" if your student ID is 108000299. Your "***StudentID*\_proj3**" folder should contain your “*algorithm\_ST.cpp”* in source folder*.*
  2. Report (10%)

1. The report file should be named "**report.pdf**."
2. Please zip your "**report.pdf**" to "**report.zip**".
3. Your project report is recommended to follow this outline:

1) Project Description

1-1) Program Flow Chart

1-2) Detailed Description

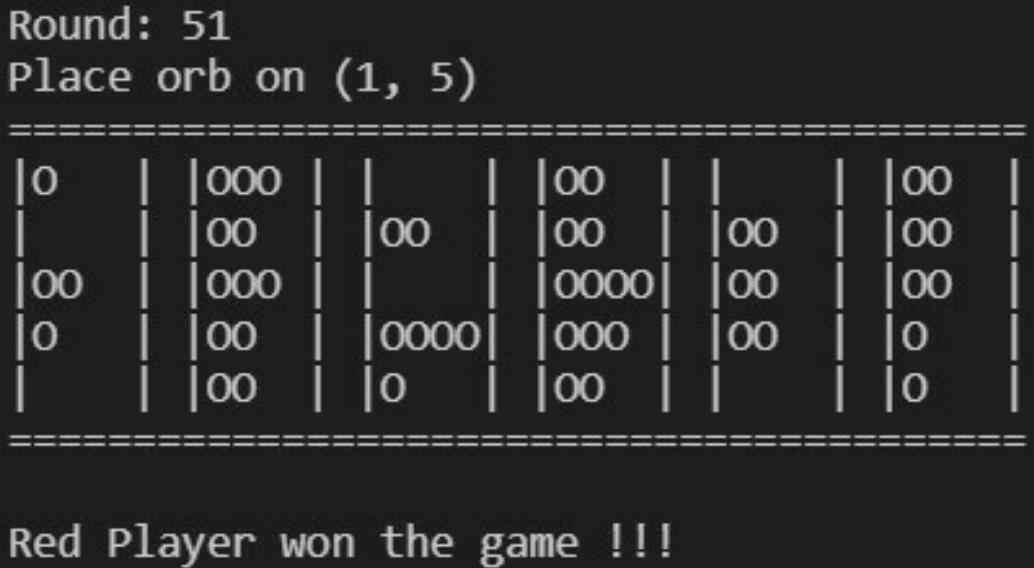
2) Screen Shots

2-1) Partial Implemented Code

2-2) GitHub Control History

2-3) Compare with TA’s AI Code (*randomMove*) for 7 results. (7 pictures)

Ex:



2-4) Describe the reason why you win TA’s AI Code or why you can’t win TA’s AI Code.

**Note:** The project report is limited to 10 pages.

**Note:** Your report can be either in Chinese or in English, or mixed.

**Etiquette**

1. **Do not plagiarize others' work, or you will fail this course.**
2. **No acceptance of late homework.**
3. **Please frequently check the class website announcements for possible updates.**