1) Project Description

1-1) Program Flow Chart

對每個格子進行分析，並做一個 5*6 的加權分數表，每個格子皆有屬於自己的分數，分數愈高者，下一步棋為該格的話，勝算愈大。而我有自己一套演算法，也就是計算分數的標準。

1-2) Detailed Description

首先，我用雙層迴圈跑過一次 5*6 上的所有格子，並分別對周圍 8 格進行比較，且計算分數。

開始對每個格子計算分數 →

A. 若此格為敵方的棋子，得到 -1000 分，並直接跑到下一個格子，因為下一步棋無法下在敵方的格子上。
B. 若此格為我方的棋子，則得到 (8 – (該格最大含棋數 – 當時棋子數))*2 分，因為遊戲規則為該格愈多棋子，愈容易爆炸並吃掉旁邊的棋子，所以同個格子上愈多我方棋子對我愈有利。
C. 若此格的周圍格子有已超過 5*6 範圍或周圍格子上無棋子，則得到 0 分。周圍格子上無棋子，對我而言不影響。
D. 若此格的周圍格子有任意棋子為我方棋子，則判斷有幾格是我方棋子，便得到幾分。因為如此只要一爆炸，周圍我方棋子即可都+1 顆，對我很有利。
E. 若此格距離爆炸的棋子數 <= 周圍格子距離爆炸的棋子數，且周圍格子為敵方的棋子，便得到 10 分。因為當我方比敵方快爆炸，即可吃掉周圍敵方的棋子，且都+1 顆，對我非常有利。
F. 若是其他狀況，則得到 -100 分。例如，此格距離爆炸的棋子數 > 周圍敵方距離爆炸的棋子數，此時若下這步棋，會容易因為敵方先爆炸而被吃掉，反而助長敵方此格的棋數，對我非常不利。

最後，將 5*6 的加權分數表中，最高分且分數 > -1000 的格子設定為下一步棋。

2) Screen Shots

2-1) Partial Implemented Code

```
34      int count[5][6] = {0};
35      int temp = -1000;
36      int color = player.get_color();
37      int opp_color;
38      if(color == 'r') opp_color = 'b';
39      else opp_color = 'r';
```

這些是我用到的變數，count[5][6]為分數加權表、color 為我方顏色、opp_color 為敵方顏色。

```
41  for(int a=0;a<5;a++){
42      for(int b=0;b<6;b++){
43          if(board.get_cell_color(a, b) == opp_color){
44              count[a][b] = -1000;
45              continue;
46          }
47          count[a][b] = (8-(board.get_capacity(a, b) - board.get_orbs_num(a, b)))*2;
```

用雙層 for 迴圈跑過每個格子一次，並計算出上述的 A、B。

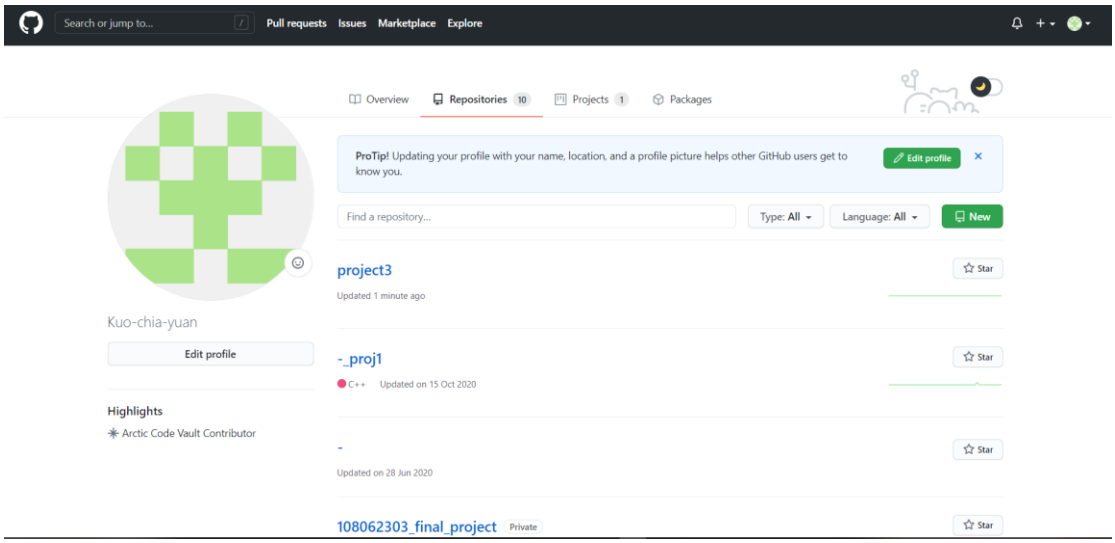
```
49          if(a-1<0 || b-1<0 || board.get_cell_color(a-1, b-1) == 'w');
50          else if(board.get_cell_color(a-1, b-1) == color){
51              count[a][b] ++;
52          }
53          else if((board.get_capacity(a, b) - board.get_orbs_num(a, b)) <= (board.get_capacity(a-1, b-1) - board.get_orbs_num(a-1, b-1))
54              count[a][b] = count[a][b] + 10;
55          }
56          else{
57              count[a][b] = count[a][b] - 100;
58          }
```

計算出上述的 C、D、E、F。並重複 8 次，因為周圍最多有 8 格。

```
138     for(int a=0;a<5;a++){
139         for(int b=0;b<6;b++){
140             if(count[a][b] > temp){
141                 temp = count[a][b];
142                 index[0] = a;
143                 index[1] = b;
144             }
145         }
146     }
147     return;
148 }
```

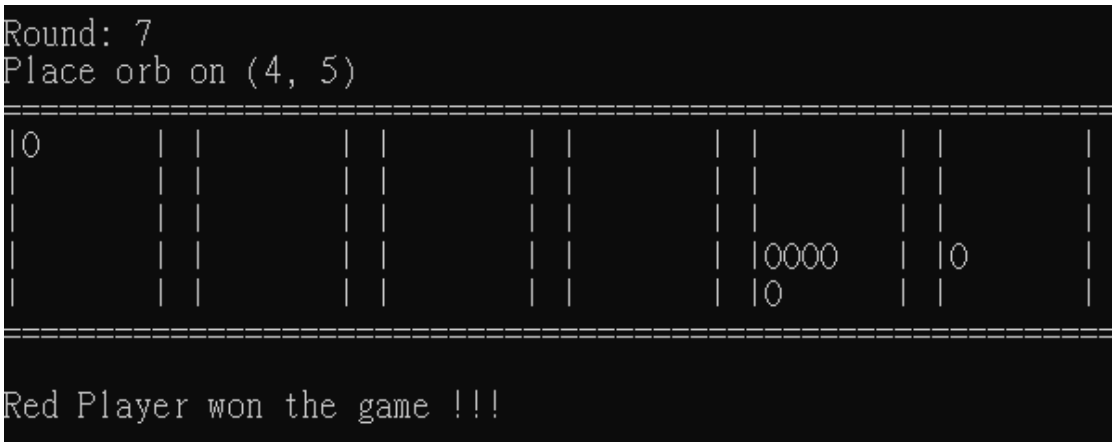
最後，將 5*6 加權分數表中，最高分且分數 > -1000 分者令為 index[0]、index[1]。

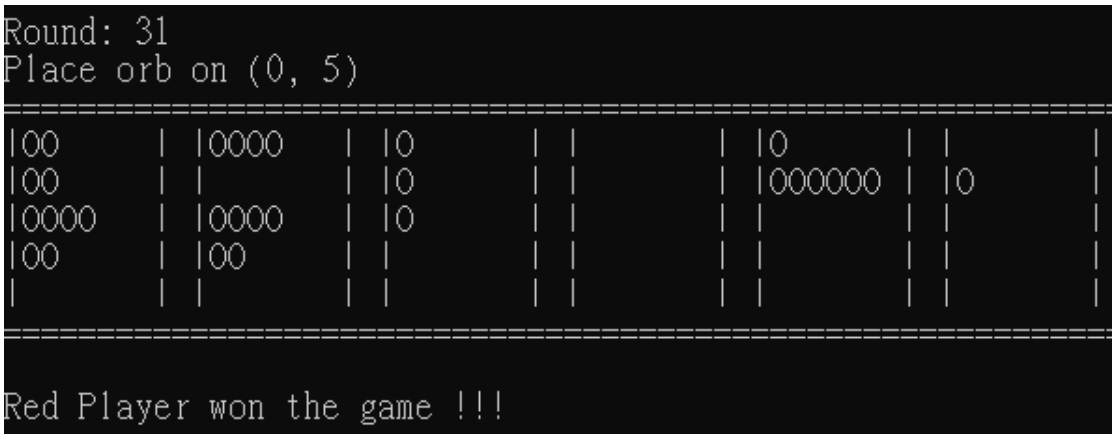
2-2) GitHub Control History

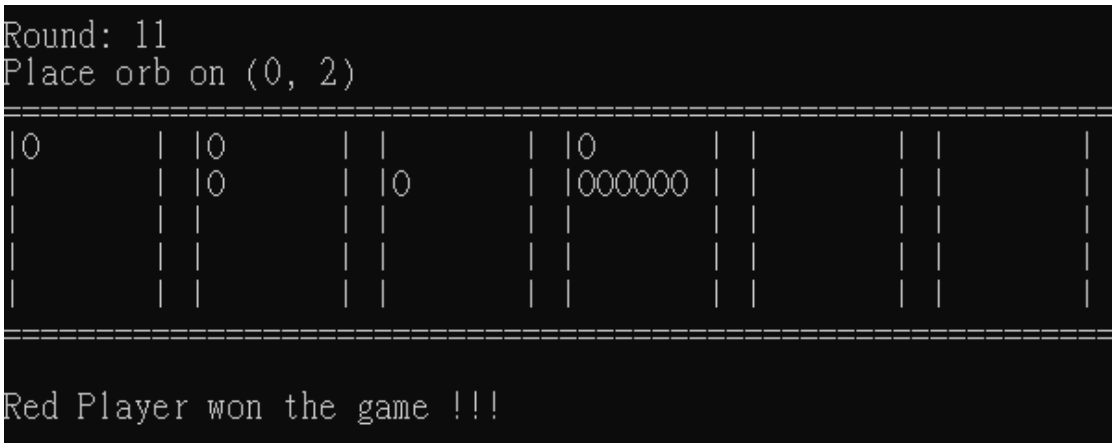


我這次遞交到 github 上的檔案為 project3。

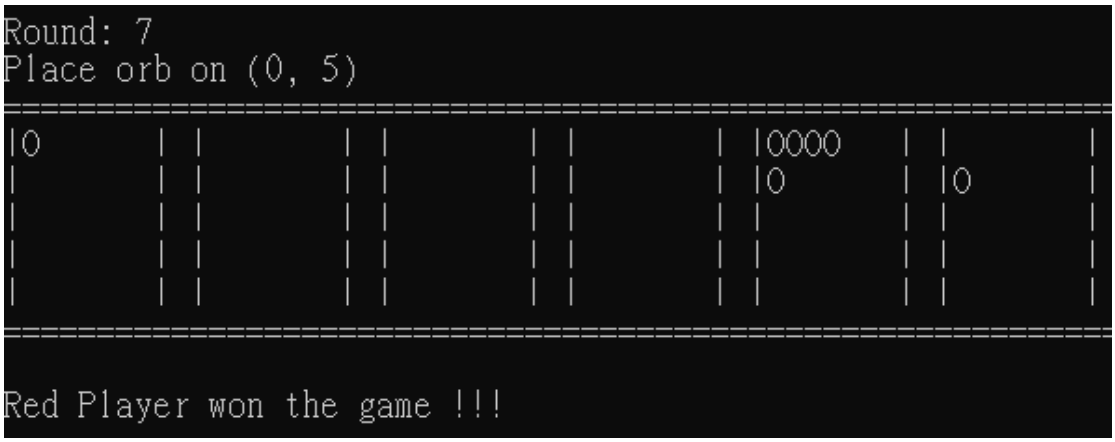
2-3) Compare with TA's AI Code (*randomMove*) for 7 results. (7 pictures)

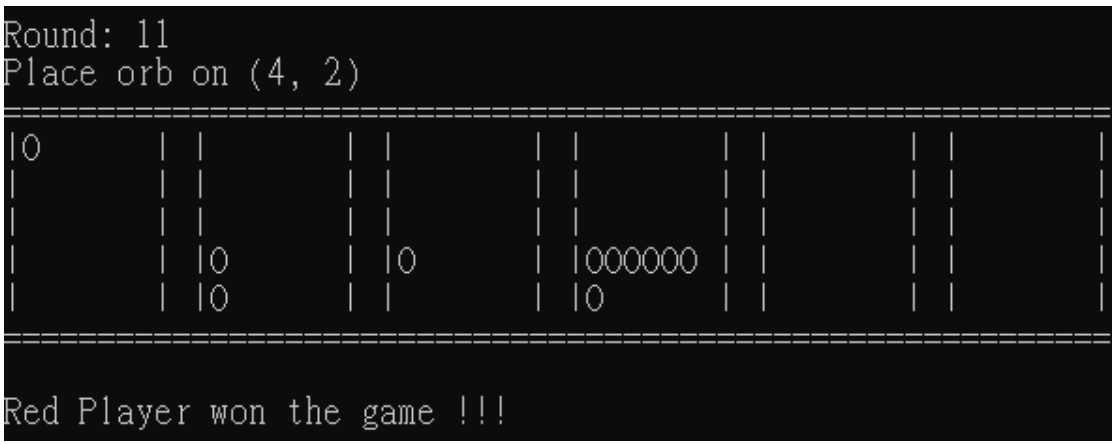





Round: 11
Place orb on (0, 2)
Red Player won the game !!!


Round: 7
Place orb on (4, 0)
Red Player won the game !!!


Round: 7
Place orb on (0, 5)
Red Player won the game !!!


Round: 11
Place orb on (4, 2)
Red Player won the game !!!

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

因為我每下一步棋，我會計算每一格的勝算，是否能吃掉敵方的棋子，或反而容易被敵方吃掉棋子。因此跟 random 下棋，不太會輸。