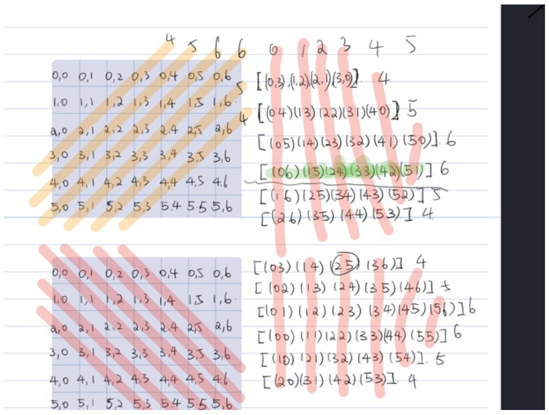
YiLun Huang
301280711

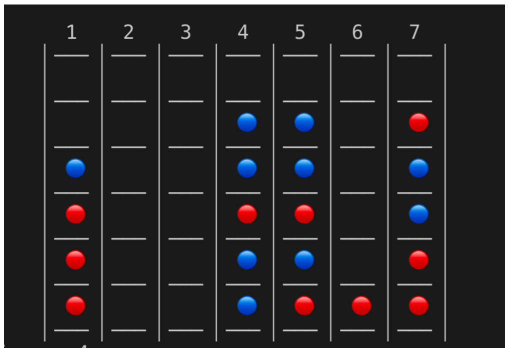
Report for Project: Connect 4

Since this is the last assignment for CMPT310, this topic is pretty opening and very challenging for me. After finishing the project, I feels horned and pleased that I used the things I learned from class and implemented to my code. I also feel excited that my program is runnable and not cause any error. I would like to make a report, start from perpetration to make conclusion.

To begin with, using Language instead of Python, I picked C++, the efficient language which can execute program much faster. In the a5.zip, there are two .h file. The Board.h and connect4.h. In the Board.h, the chess board is being initialized when call the constructor, and also, I use the deep copy function, that when I need to pass all the current board information to another new board. Destructor to make sure that everything is clean up, this searching program can be wasted of memory. The board is initially having all 0's, 0 represent empty spot. 1 represent red chess and 2 represent blue. And there is list keep tracking of all valid position, initially to 0, when add one chess in there, then the corresponding index will increase by 1. This is a very important index which will tell me which role that I can put the chess in. There are three more data: row list, column list, and diagonal list. For a 6 x 7 table, to make a diagonal list, which have all the correct value of spot is very hard. So I did the hard code, which transfer each of the spot from 2-D array into diagonal list. There are shown down below



Convert row list to column list, copy all the data from one to another is easier than diagonal list. After implementation of Board.h, then I can use this board, insert chess to the table and print it out.



YiLun Huang
301280711

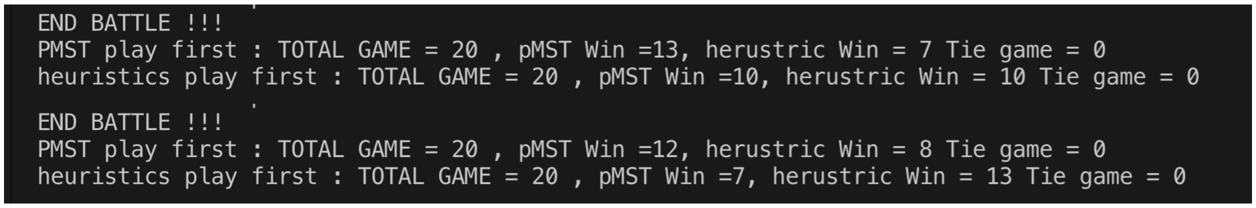
Tried as much as I can to duplicate the board, but the chess color is different. Thanks for the Unicode. Print the chess board is not only for player to play, but also for me debugging during implementation.

After finished all implementation of the basic function for board class, then I created a AI.cpp which is the old version of connect4.cpp, I combine the AI part and the main function together into one file.

Since Assignment 4 is doing the pMCT for Tic-Tac-Toe, So I start implement the pMCT for connect four. The basic concept is the same. Instead of doing 500 for Tic-Tac-Toe using Python, my default random time are 1000 times, which is really faster in c++. The AI will be taking all the valid positions from the current board using for loop, to check the grade for each position. Win +5 and Loss -3 and Tie +0. Found out that when random simulation for 1000 times and the grade calculation is perform good. In one random simulation, the AI will first make the pick move from the valid position, and then start the real simulation. Which can be understand as when I take this move, how many points will I get or what my grade of taking this move. Then the computer starts doing random simulation, by using system clock, srand (time (NULL)). Finally find the maximum points from the grade list and find the corresponding move. This is good at most of the cases. The pMCT AI is not perfect. There are much be some brunches being missed.

About for the heuristic AI, I choose using Minimax searching algorithm. And I'm trying to make the depth of searching to 5, but the outcome does not show correctly, so instead of choosing 5, I choose 2. Which means the AI will consider when taking the chosen move, what if the opponent makes this move. Which is the main idea of Minimax algorithm. There are several situations that need to define. When make this move, if there are connect of four, then which means win, it will return +10000, and when connect of three, it will return +5, and when connect of two, +2. Also, when It put to the central of the board, which is pos = 4, will give additional +4. It encouraged AI to take the central bar to make advantage. Then when others make connect of four, then directly return -100, which means this is the worst situation, and when opponent can make connect of three, -50, when make connect 2, -2. This play good in most of the situation. The only disadvantage is the depth need to be higher.

There are three game mode, the last one is watching these two AI have a battle. The order of which AI going first is the essential points.



```
END BATTLE !!!
PMST play first : TOTAL GAME = 20 , pMST Win =13, herustric Win = 7 Tie game = 0
heuristics play first : TOTAL GAME = 20 , pMST Win =10, herustric Win = 10 Tie game = 0

END BATTLE !!!
PMST play first : TOTAL GAME = 20 , pMST Win =12, herustric Win = 8 Tie game = 0
heuristics play first : TOTAL GAME = 20 , pMST Win =7, herustric Win = 13 Tie game = 0
```

YiLun Huang
301280711

heuristics go first

#Win	#Loss	#Tie	Win Ratio
10	10	0	0.5
10	10	0	0.5
13	7	0	0.65
11	9	0	0.55
11	9	0	0.55

heuristics go second

#Win	#Loss	#Tie	Win Ratio
9	9	2	0.45
7	13	0	0.35
8	12	0	0.4
6	11	3	0.3
5	15	0	0.25

From the above table, it shown that the order of playing this game is essential, when heuristics AI playing first, the wining ration is around or above 60%, but when pMCT playing first, the wining ratio are dropping to below 40% or less.

In conclusion, who choose to play first has a huge advantage.