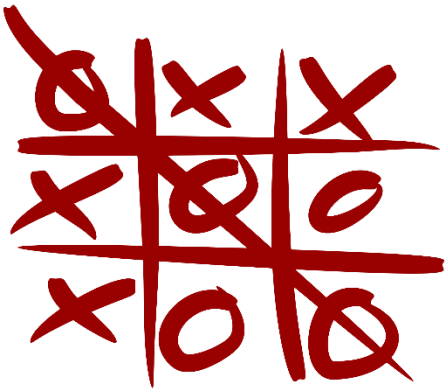
Recently, AlphaGo got its reputation by beating a professional player at the game of Go. It is widely viewed as a milestone for development of artificial intelligence. Here, we are going to design a program to play the game of XO, or “井字棋” in Chinese. Different from the 19×19 board in the game of the Go, two players, X and O, in the game of XO will take turns marking the spaces in a 3×3 grid. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row wins the game.



In this project, you need to complete the game with 2 human players. OOP techniques is encouraged in the program design. The following classes may be designed and implemented in your program.

class **game**: to manage the game.
class **board**: to describe the 3×3 grid board and the rules of the game. Please protect the data to prevent the players make a modification arbitrarily.
class **player**: there should be two players as class members in the class **game**, a virtual function in the class design is suggested.
class **playerHuman**: derived from base class **player**.

The grade of the project is based on the correction of your program, the quality of the report and the friendliness of the user interface.

The implementation of a computer player in the game of XO would have a bonus. You can use class **playerComputer**, which is derived from base class **player**, in the program design.

Hint:
Combining Monte-Carlo tree search with deep neural networks, AlphaGo is implemented by a "value network" and a "policy network". In the 3×3 grid of the XO game, we can use just one layer of value network in our program design. In fact, Newell and Simon in 1970s already developed a perfect strategy for the XO game, which are realized as follows,

1. **Win**: If the player has two in a row, they can place a third to get three in a row.
2. **Block**: If the opponent has two in a row, the player must play the third themselves to block the opponent.

3. **Fork**: Create an opportunity where the player has two threats to win (two non-blocked lines of 2).
4. **Blocking an opponent's fork**:
    - **Option 1**: The player should create two in a row to force the opponent into defending, as long as it doesn't result in them creating a fork. For example, if "X" has a corner, "O" has the center, and "X" has the opposite corner as well, "O" must not play a corner in order to win. (Playing a corner in this scenario creates a fork for "X" to win.)
    - **Option 2**: If there is a configuration where the opponent can fork, the player should block that fork.
5. **Center**: A player marks the center.
6. **Opposite corner**: If the opponent is in the corner, the player plays the opposite corner.
7. **Empty corner**: The player plays in a corner square.
8. **Empty side**: The player plays in a middle square on any of the 4 sides.