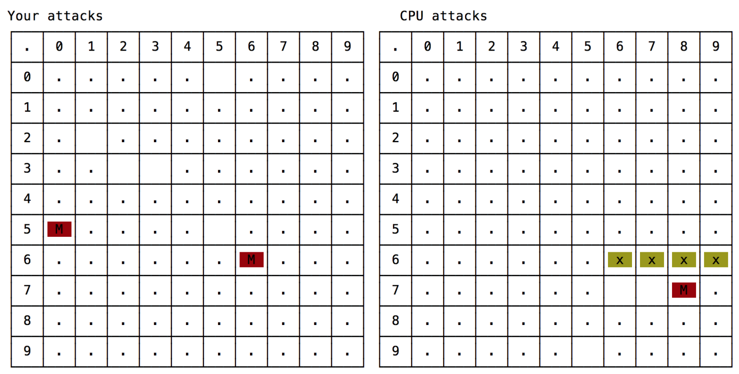
Homework III: 2D Battleship Bot

.

Battleship (also Battleships or Sea Battle) is a guessing game for two players. It is known worldwide as a pencil and paper game which dates from World War I. It was published by various companies as a pad-and-pencil game in the 1930s, and was released as a plastic board game by Milton Bradley in 1967.
The game is played by two players. Each player has typically a grid that is 10x10. The rows on that grid are identified by a number, whereas the columns are identified by a letter. Before you play the game, each player chooses where his ships are arranged.

The game is played in rounds; in each round each player takes a turn attacking a target square in the opponent's grid by choosing a coordinate (row, col). If a ship occupies that square the player announces that his ship has been hit, otherwise it was a miss.



Your attacks

.	0	1	2	3	4	5	6	7	8	9
0	.	.	.	.	.		.	.	.	.
1	.	.	.	.	.	.	.	.	.	.
2	.		.	.	.	.	.	.	.	.
3	.	.			.	.	.	.	.	.
4	.	.	.	.	.	.	.	.	.	.
5	M	.	.	.	.		.	.	.	.
6	.	.	.	.	.	.	M	.	.	.
7	.	.	.	.	.	.	.	.	.	.
8	.	.	.	.	.	.	.	.	.	.
9	.	.	.	.	.	.	.	.	.	.

CPU attacks

.	0	1	2	3	4	5	6	7	8	9
0	.	.	.	.	.	.	.	.	.	.
1	.	.	.	.	.	.	.	.	.	.
2	.	.	.	.	.	.	.	.	.	.
3	.	.	.	.	.	.	.	.	.	.
4	.	.	.	.	.	.	.	.	.	.
5	.	.	.	.	.	.	.	.	.	.
6	.	.	.	.	.	.	x	x	x	x
7	.	.	.	.	.	.			M	.
8	.	.	.	.	.	.	.	.	.	.
9	.	.	.	.	.		.	.	.	.

1 Your task:

Now that you are seasoned vets, with an understanding of control structures, functions and repetitions; your task is to implement a computer Bot that can play a 2D battleship game.

A Bot (short for "Robot") is a computer program that operates as an agent for a user or another program or simulates a human activity (definition taken from `http://searchsoa.techtarget.com/definition/bot`). The battleship Bot job is to decide in each turn which location of the opponent's gird it should fire upon. The Bot can take into consideration previous historical data that it has gathered from previous plays in addition to some additional input which we will discuss later.

This task is not about implementing the 2D battleship game but rather implementing a simplified artificial intelligence (AI) program that can play and win 2D battleship. The target of your Bot is to win the game by sinking your opponent's ship before he sinks yours.

2 Starter code:

Under the assignment tab on NYUclasses, you will download the BattleOfTheBots starter code. This includes two files: BattleshipBot.py and network.pyc. The network.pyc file is a special module/library that we have created for you, which will help your Bot in playing against other bots or even against a computer AI. The BattleshipBot.py is the started code for your homework. It includes the necessary starting code structure for you to use to start implementing your Bot.

2.1 network.pyc

- **network.createBoards(name [, True])**: this method is responsible for creating your 2D battleship board and connecting your program with the BattleOfTheBots server. The method takes two arguments, the first argument is a mandatory argument which is the name of your Bot. The second argument is optional, if defined to be True then your Bot will be playing a game against the computer. This is to help you debug you code, so that you can test your functionality and implementation.
  The function will return four values back these are:
  - dispBoard: this is a 2D board that is filled with '.'s. This you can use for displaying your opponent's board with your guesses.
  - size: this is an integer value specifying your board size. By default its 10, i.e., the board is going to be of size 10x10.
  - shipSize: this is an integer value that specifies the size of your ship, i.e., how many locations does your ship occupy. By default your ship consists of 4 locations, and the ship direction is randomly picked either vertically of horizontally.
  - mines: this is an integer value specifying how many mines are there in the board. By default this value has been set to 25.
- **network.receive()**: this method is responsible for receiving the opponents play coordinates. This is used to give your opponents a chance to fire at one location from within your board. This method does not take any arguments. It returns however a value which can take one of the following values:
  - 'lost': if this is received it means that you have lost the game, i.e., your opponent has shot your last ship location and has sunk your ship.
  - None: this is returned in case your opponent has hit a location in your board which is either an empty slot or one of your ship locations
  - 'r,c': a string that contains two integer values separated by a comma, e.g. '0,5'. This specifies a near by location of your opponent's ship. If your opponent hits one of your mines, you will be given additional information on the vicinity of

the opponent’s ship. This location is a randomly chosen slot which is one block away from the opponents ship. You can use this information as additional input/hint for your Bot to figure out where to hit next.

- **network.send(guess)**: this method is responsible for sending your hit coordinates to the opponent. It takes one string argument ”r,c” which consists of two integer values separated by comma, where the first one represents the row you want to hit and the second represents the column you want to hit. This method returns one of the following values:
  - 'win': if the coordinate that you have sent is the last location of your opponent’s ship, it means that you have sunk his ship and you have won the game.
  - 'wrong input': if the position that you have sent is a wrong position. For example, if you send coordinates that are out of the board’s range, or that are not integers but strings.
  - one of the following strings (' ', 'x', 'M'): the answer of what value was behind the location that you have hit. This can be an empty place (' '), or a ship location ('x'), or a mine 'M'.

2.2 BattleshipBot.py

```python
import network
import random,time,os

name = input ("Please enter your name")

# create boards
dispB, size, shipSize, mines = network.createBoards(name, True)

while True:
        ans = network.receive()

        if ans == 'lost':
                break
        elif ans != None:
                ans = list(map(int,ans.split(',')))
                print (ans[0],ans[1], "is nearby location of the ship")

        guess = input("enter coordinates r,c: ")

        ans = network.send(guess)

        if ans == 'win':
                break
        elif ans == 'wrong input':
                print ("Your input was wrong")
        else:
                # assign result to board
                guessList = list(map(int,guess.split(',')))
                dispB[guessList[0]][guessList[1]]=ans


# Printing final result
if ans == 'win':
        dispBoard[guessList[0]][guessList[1]]='x'
        network.printBoard()
        print ("you have won the game")
else:
        network.printBoard()
        print ("you have lost the game")
```

At the beginning of the starter code (line 1-2), we import the relevant modules that we will use (network, random, time, os). Remember that you have to copy the network.pyc file into the same folder where your starter code will be.

In line 7, we create our board and connect to the server, notice there that we are giving the second optional argument to be True. This means that we want to play the game against the computer.

line 9 to 29 is the main game loop. We start it first by waiting for the opponents move in line 10. Depending on the answer that we will get, we either lose the game or we might get an indication on where the opponent's ship lies in case the opponents hits one of our mines (line 12-16). Notice that your Bot can use this information to decide where to hit next. The ans variable in line 15 will have a one block away location of your opponent's ship.

line 18, your Bot code should kick in here. At the moment we are asking the user for input on where the next hit needs to be. You need to change this. This is where your Bot logic will be implemented. The main outcome of your logic is to come up with a string 'r,c' that represents the location on where you want to hit next.

This string is then used in line 20, and is sent to the network. Based on what is behind this location, you will either win, in case you hit all 4 ship locations, or you will get the value behind that location which you will need to assign to your display board (line 27-29). In case you gave the wrong location, you will get the 'wrong input' string (line 25-25).

Upon exiting the main game loop we will decide based on the ans whether we won the game or not (line 32-39)