Homework 3

Moodle Submission Deadline: 2017/11/26 (Sunday) 23:59

[Bring the printed papers to the class on **11/28** (Tue)]

Problem 1: Tic-Tac-Toe Game (hw3_p1.py)

Tic-Tac-Toe (a.k.a. **Xs and Os** 井字遊戲) is a game for two players, X and O, who take turns marking the cells 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. The following example game is won by the player X:

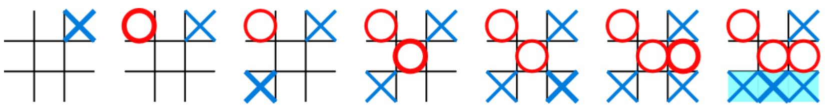


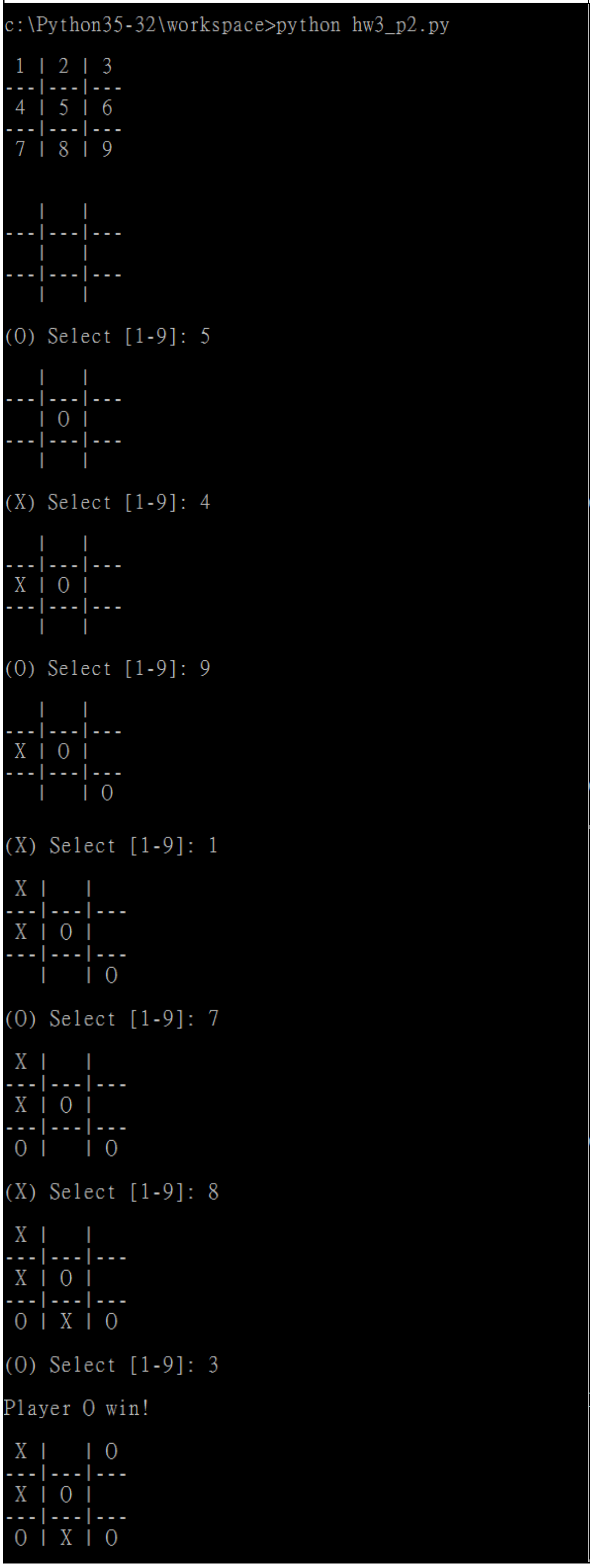
Image from Wikipedia https://en.wikipedia.org/wiki/Tic-tac-toe

Your task is to provide the controlling logic and the text-based presentation for a two-player Tic-Tac-Toe game. Since it is a two-player game, your program should allow two players X and O to interact with each other in the game. To help you understand the game logic in the programming, we divide the problem into six of the division of labor between the program logic and the presentation, as shown in the following table. The program logic forms the game engine, enforcing the rules of the game. The presentation is the text-based interface to the player (keyboard input and text output).

What should your program do?	Action and Presentation
(1) Inform both players about the positions [1-9] of the board at the beginning of the game.	Draw a 3×3 board that shows position numbers [1-9] in the corresponding cells of the board. 1 | 2 | 3 4 | 5 | 6 7 | 8 | 9
(2) Keep track of whose turn it is.	Indicate the current player (X or O).
(3) Allow a player to select a cell [1-9] in the board at each round. Make sure disallow illegal moves.	A player can select a cell from the board by entering 1-9 at each round. A cell that had been selected cannot be selected again. In addition, provide feedback to the player about an illegal selection: ask the player to enter the selection again until a legal selection is entered.
(4) Place selected marks (X or O) in the specified position of the board.	Draw the selections (Xs and Os) from the beginning to the current round. Let those cells without selection be blank.
(5) Implement the winning rule.	For a player, if there are three of his/her marks (selections) in a horizontal, vertical, or diagonal row, he/she wins the game.
(6) Determine whether or not there is a winner at each round.	If there is a winner, show the winner, show the board, and terminate the game. If there is no winner, simply show the current board to let the players know the current selections.

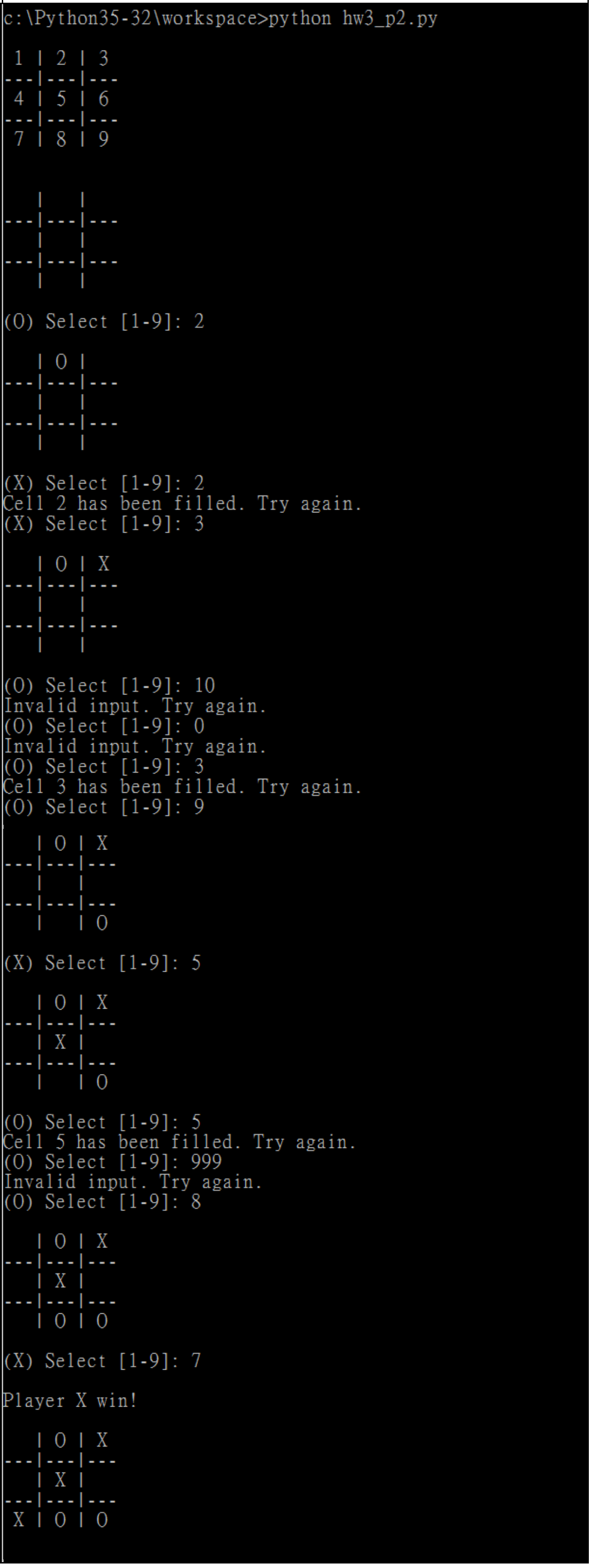
Sample Input and Output

Example 1



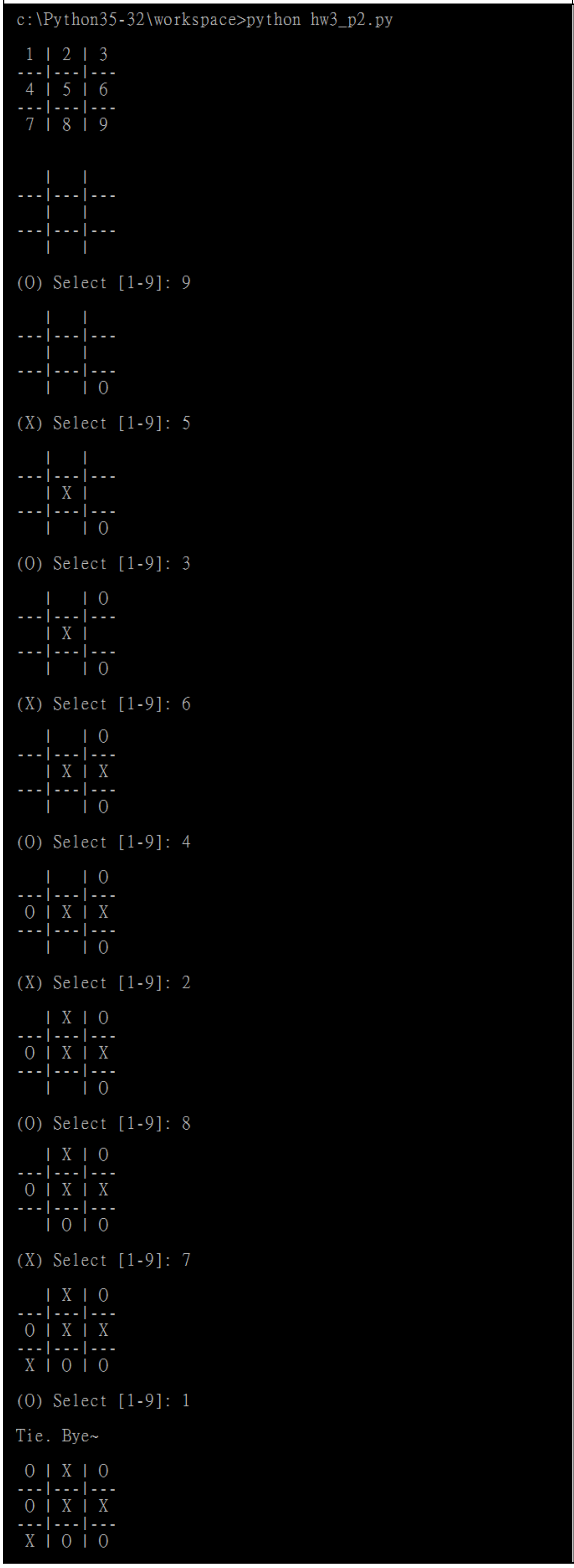
```
c:\Python35-32\workspace>python hw3_p2.py

 1 | 2 | 3
---|---|---
 4 | 5 | 6
---|---|---
 7 | 8 | 9


   |   |
---|---|---
   |   |
---|---|---
   |   |

(O) Select [1-9]: 5

   |   |
---|---|---
   | O |
---|---|---
   |   |

(X) Select [1-9]: 4

   |   |
---|---|---
 X | O |
---|---|---
   |   |

(O) Select [1-9]: 9

   |   |
---|---|---
 X | O |
---|---|---
   |   | O

(X) Select [1-9]: 1

 X |   |
---|---|---
 X | O |
---|---|---
   |   | O

(O) Select [1-9]: 7

 X |   |
---|---|---
 X | O |
---|---|---
 O |   | O

(X) Select [1-9]: 8

 X |   |
---|---|---
 X | O |
---|---|---
 O | X | O

(O) Select [1-9]: 3

Player O win!

 X |   | O
---|---|---
 X | O |
---|---|---
 O | X | O
```

Example 2



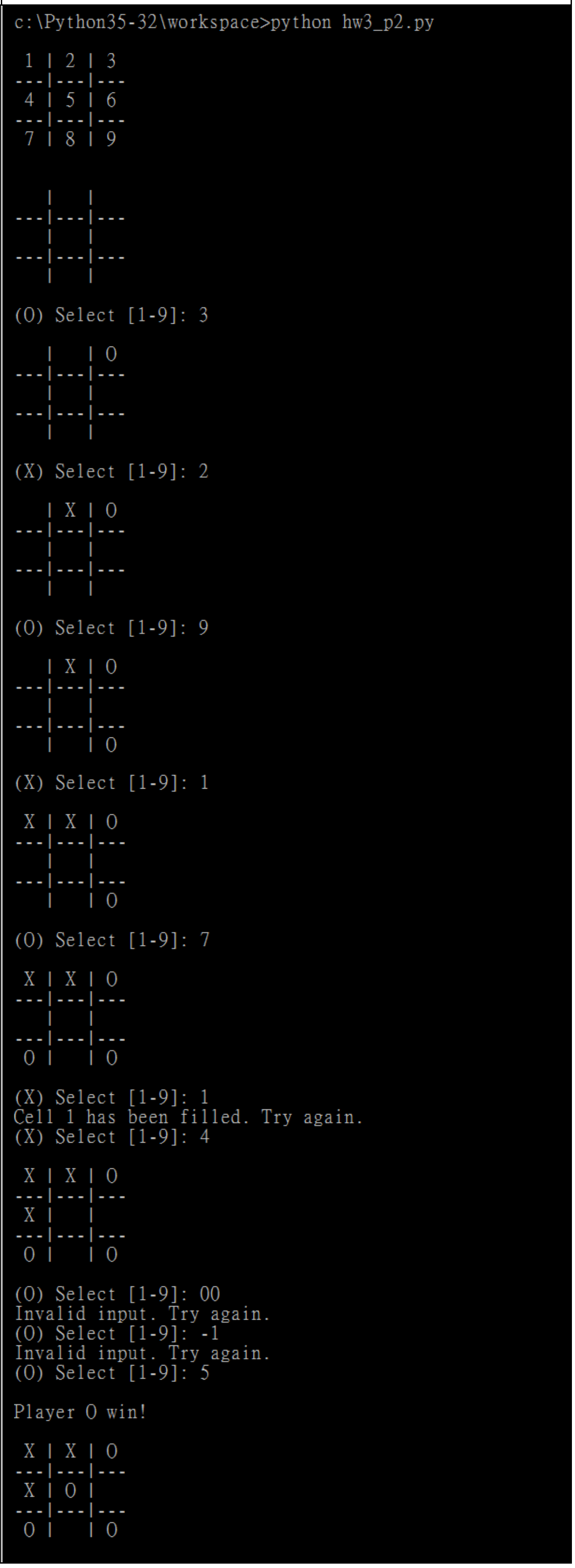
```
c:\Python35-32\workspace>python hw3_p2.py

 1 | 2 | 3
---|---|---
 4 | 5 | 6
---|---|---
 7 | 8 | 9


   |   |
---|---|---
   |   |
---|---|---
   |   |

(O) Select [1-9]: 2

   | O |
---|---|---
   |   |
---|---|---
   |   |

(X) Select [1-9]: 2
Cell 2 has been filled. Try again.
(X) Select [1-9]: 3

   | O | X
---|---|---
   |   |
---|---|---
   |   |

(O) Select [1-9]: 10
Invalid input. Try again.
(O) Select [1-9]: 0
Invalid input. Try again.
(O) Select [1-9]: 3
Cell 3 has been filled. Try again.
(O) Select [1-9]: 9

   | O | X
---|---|---
   |   |
---|---|---
   |   | O

(X) Select [1-9]: 5

   | O | X
---|---|---
   | X |
---|---|---
   |   | O

(O) Select [1-9]: 5
Cell 5 has been filled. Try again.
(O) Select [1-9]: 999
Invalid input. Try again.
(O) Select [1-9]: 8

   | O | X
---|---|---
   | X |
---|---|---
   | O | O

(X) Select [1-9]: 7

Player X win!

   | O | X
---|---|---
   | X |
---|---|---
 X | O | O
```

Example 3



```
c:\Python35-32\workspace>python hw3_p2.py

 1 | 2 | 3
---|---|---
 4 | 5 | 6
---|---|---
 7 | 8 | 9


   |   |
---|---|---
   |   |
---|---|---
   |   |

(O) Select [1-9]: 9

   |   |
---|---|---
   |   |
---|---|---
   |   | O

(X) Select [1-9]: 5

   |   |
---|---|---
   | X |
---|---|---
   |   | O

(O) Select [1-9]: 3

   |   | O
---|---|---
   | X |
---|---|---
   |   | O

(X) Select [1-9]: 6

   |   | O
---|---|---
   | X | X
---|---|---
   |   | O

(O) Select [1-9]: 4

   |   | O
---|---|---
 O | X | X
---|---|---
   |   | O

(X) Select [1-9]: 2

   | X | O
---|---|---
 O | X | X
---|---|---
   |   | O

(O) Select [1-9]: 8

   | X | O
---|---|---
 O | X | X
---|---|---
   | O | O

(X) Select [1-9]: 7

   | X | O
---|---|---
 O | X | X
---|---|---
 X | O | O

(O) Select [1-9]: 1

Tie. Bye~

 O | X | O
---|---|---
 O | X | X
---|---|---
 X | O | O
```

Example 4



```
c:\Python35-32\workspace>python hw3_p2.py

 1 | 2 | 3
---|---|---
 4 | 5 | 6
---|---|---
 7 | 8 | 9


   |   |
---|---|---
   |   |
---|---|---
   |   |

(O) Select [1-9]: 3

   |   | O
---|---|---
   |   |
---|---|---
   |   |

(X) Select [1-9]: 2

   | X | O
---|---|---
   |   |
---|---|---
   |   |

(O) Select [1-9]: 9

   | X | O
---|---|---
   |   |
---|---|---
   |   | O

(X) Select [1-9]: 1

 X | X | O
---|---|---
   |   |
---|---|---
   |   | O

(O) Select [1-9]: 7

 X | X | O
---|---|---
   |   |
---|---|---
 O |   | O

(X) Select [1-9]: 1
Cell 1 has been filled. Try again.
(X) Select [1-9]: 4

 X | X | O
---|---|---
 X |   |
---|---|---
 O |   | O

(O) Select [1-9]: 00
Invalid input. Try again.
(O) Select [1-9]: -1
Invalid input. Try again.
(O) Select [1-9]: 5

Player O win!

 X | X | O
---|---|---
 X | O |
---|---|---
 O |   | O
```

Hints for Problem 1 (僅為參考，不代表只有這種作法)

(1) 用一個長度為 9 的 list 來儲存 board，初始化該 list 為 9 個-1 來代表這 9 格都尚未被選，若被 O 選，則將對應號碼格子之 list 位置設為 0，反之若被 X 選，則設為 1。

(2) 用一個 nested list，該 list 內存放 8 個 lists，代表 8 種能使某玩家贏得遊戲的連線位置，例如：`[[1,2,3],[4,5,6],[7,8,9],[1,4,7],...]`(依此類推)，用此 nested list 於每一回合檢查是否有某玩家已滿足其中一連線；若是，則該玩家勝利，若否，遊戲繼續。

(3) 此題主要練習以及熟練使用 list，其他可能會用到多層迴圈、`continue`、`break`、`if-elif-else` 條件判斷式。

(4) 可使用不換行之列印，如 `print('O', end="")`，來將 O 與 X 來填入並印出 board。

Problem 2: Find the Best Players in NBA History (hw3_p2.py)

The National Basketball Association (NBA) is the USA's premier professional men's basketball league. The NBA has 30 teams; 29 in the United States and one in Canada. The NBA is one of the four major North American professional sports leagues which are NBA, NHL, NFL and MLB. Now you are asked to write a program that can find the best players in NBA history. How do many NBA coaches quickly evaluate a player's game performance? They check the *efficiency* scores of players.

The problem aims to let you practice how to smartly store and process text files. You will practice the manipulation of either the `list` data structure. **We provide you a *csv* file (nba_data.csv) that contains the raw NBA statistic data of all the players from 1946 to 2009. You are asked to write a program to compute the *career efficiency* for all the players, and output the 20 highest efficiency players in terms of *career efficiency*.**

The format of **nba_data.csv** is easy to understand. The first line tells you the names of all columns. From the second line, each line's data corresponds to one player's different field values in a **particular** regular season. For example, suppose Michael Jordan had played multiple seasons, his data contains multiple lines for his seasons. You should note that if Michael Jordan had played for two different teams (for example: Chicago Bulls and LA Lakers) in season 1990, there will be two lines for his data in season 1990, in which two lines correspond to his data in Bulls and Lakers respectively. In this case, your program should first sum up the value of each column to obtain the total value in a season. For example, suppose Michael Jordan made 1800 points (pts) in Chicago Bulls in season 1990 and made

900 points (pts) in LA Lakers in season 1990. Then his total point made in season 1990 is 2700.

The efficiency of a player in a regular season i, termed eff_season_i, is defined by this formula.

$$eff\_season\_i = \frac{(pts + reb + asts + stl + blk) - ((fga - fgm) + (fta - ftm) + turnover)}{gp}$$

You can find the technical words on the right-hand side of this formula in the first line of **nba_data.csv**. Note that you can check out the meanings of each of the abbreviations in the end of this assignment. Since the above eff_season_i is for a particular regular season i, you need to compute the player's efficiency for every season, and average them to obtain the *career efficiency* of the player, termed eff_career. In other words, you can obtain eff_career by this formula:

$$eff\_career = \frac{\sum_{i=1}^{n} eff\_season\_i}{n}$$

where n is the number of seasons that the player played.

When you find the 20 players with the highest 20 career efficiency scores, you are asked to rank these players according to their career efficiency scores. Players with higher efficiency scores are ranked at top positions. Then you need to write these players into the output file **nba_best.txt**. In the output file, each line is required to have the format:

```
Rank X\tfirstname lastname\tcareer_efficiency
```

Sample Output:

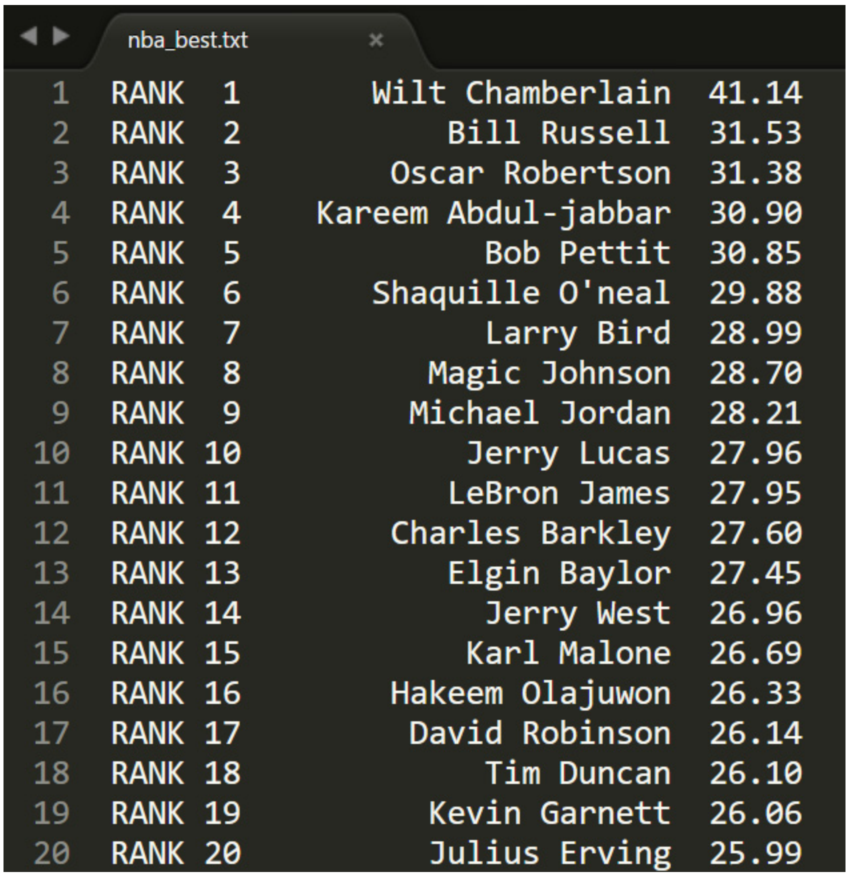
```
RANK  1      Wilt Chamberlain    41.14
RANK  2          Bill Russell    31.53
RANK  3       Oscar Robertson    31.38
RANK  4   Kareem Abdul-jabbar    30.90
RANK  5            Bob Pettit    30.85
...
```

Abbreviations (in nba_data.csv)

- gp - Games Played (打了場數)
- minutes - Minutes Played (打了分鐘數)
- pts - Points made (得分)
- reb - Total Rebounds (籃板)
- asts - Total Assists (助攻)
- stl - Steals (抄截)

- blk - Blocks (火鍋)
- fgm - Field Goals Made (跳投命中次數)
- fga - Field Goals Attempted (跳投出手次數)
- ftm - Free Throws Made (罰球命中次數)
- fta - Free Throws Attempted (罰球出手次數)
- turnovers (失誤)

Output File (nba_best.txt)


nba_best.txt

```
RANK  1       Wilt Chamberlain  41.14
RANK  2           Bill Russell  31.53
RANK  3       Oscar Robertson   31.38
RANK  4    Kareem Abdul-jabbar  30.90
RANK  5             Bob Pettit  30.85
RANK  6      Shaquille O'neal   29.88
RANK  7             Larry Bird  28.99
RANK  8         Magic Johnson   28.70
RANK  9         Michael Jordan  28.21
RANK 10            Jerry Lucas  27.96
RANK 11           LeBron James  27.95
RANK 12       Charles Barkley   27.60
RANK 13           Elgin Baylor  27.45
RANK 14             Jerry West  26.96
RANK 15            Karl Malone  26.69
RANK 16       Hakeem Olajuwon   26.33
RANK 17         David Robinson  26.14
RANK 18             Tim Duncan  26.10
RANK 19         Kevin Garnett   26.06
RANK 20          Julius Erving  25.99
```

[Bonus] **Problem 3: Candy Crush** (**hw3_p3.py**) [可交可不交]

This problem is about implementing a basic elimination for Candy Crush. Given a 2D integer array `board` representing the grid of candy, different positive integers `board[i][j]` represent different types of candies. A value of `board[i][j] = 0` represents that the cell at position `(i, j)` is empty. The given board represents the state of the game following the player's move. Now, you need to restore the board to a stable state by crushing candies according to the following rules:

(1) If three or more candies of the same type are **adjacent vertically or horizontally**, **"crush" them all at the same time** - these positions become empty.

(2) After crushing all candies simultaneously, if an empty space on the board has candies on top of itself, then these candies will drop until they hit a candy or bottom at the same time. (No new candies will drop outside the top boundary.)

(3) After the above steps, there may exist more candies that can be crushed. If so, you need to repeat the above steps.

(4) If there does not exist more candies that can be crushed (ie. the board is stable), then return the current board.

You need to perform the above rules until the board becomes stable, then return the current board.

Sample Input/Output: (read from candy_input.txt, output to candy_output.txt)

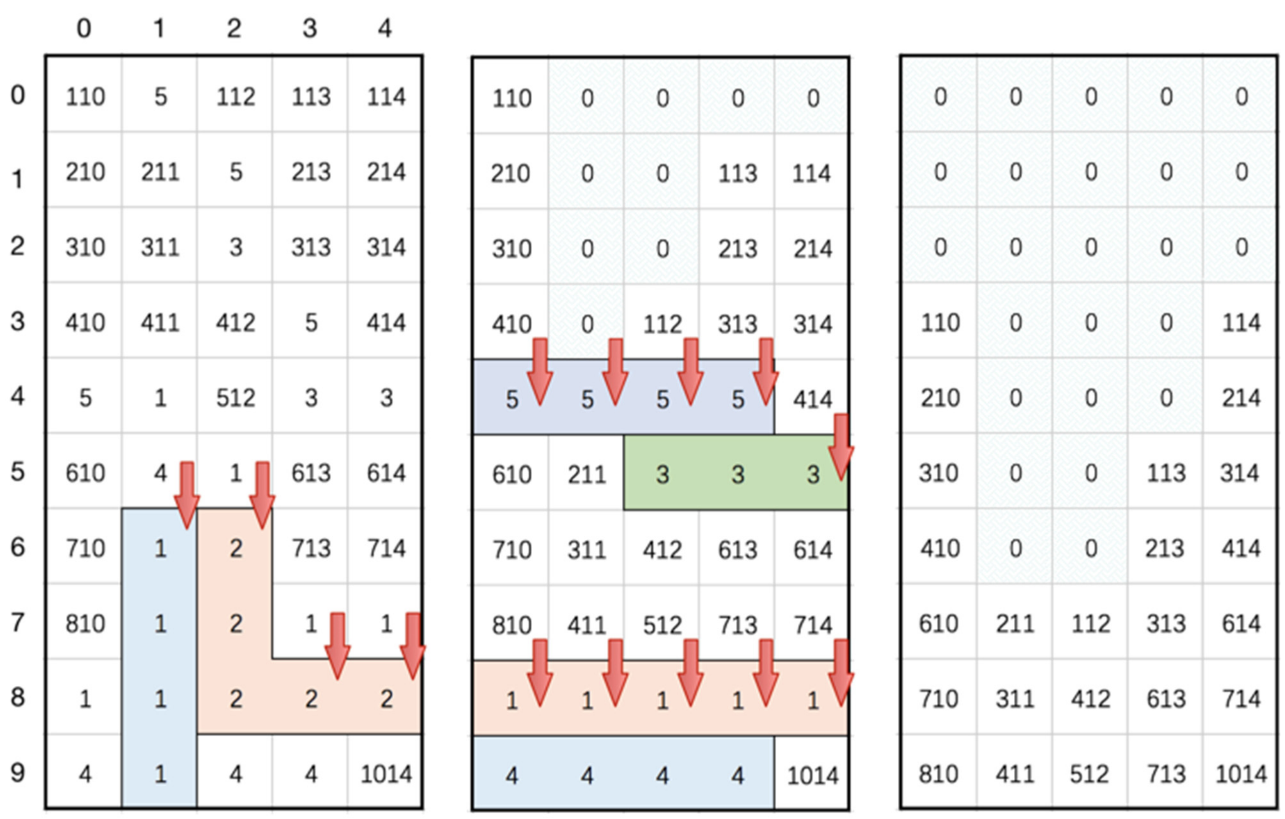
Example Input 1:

[[110,5,112,113,114],[210,211,5,213,214],[310,311,3,313,314],[410,411,412,5,414],[5,1,512,3,3],[610,4,1,613,614],[710,1,2,713,714],[810,1,2,1,1],[1,1,2,2,2],[4,1,4,4,1014]]

Example Output 1:

[[0,0,0,0,0],[0,0,0,0,0],[0,0,0,0,0],[110,0,0,0,114],[210,0,0,0,214],[310,0,0,113,314],[410,0,0,213,414],[610,211,112,313,614],[710,311,412,613,714],[810,411,512,713,1014]]

Step-by-step Example 1:



	0	1	2	3	4
0	110	5	112	113	114
1	210	211	5	213	214
2	310	311	3	313	314
3	410	411	412	5	414
4	5	1	512	3	3
5	610	4	1	613	614
6	710	1	2	713	714
7	810	1	2	1	1
8	1	1	2	2	2
9	4	1	4	4	1014

Candy crush and Drop

110	0	0	0	0
210	0	0	113	114
310	0	0	213	214
410	0	112	313	314
5	5	5	5	414
610	211	3	3	3
710	311	412	613	614
810	411	512	713	714
1	1	1	1	1
4	4	4	4	1014

Candy crush and Drop

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
110	0	0	0	114
210	0	0	0	214
310	0	0	113	314
410	0	0	213	414
610	211	112	313	614
710	311	412	613	714
810	411	512	713	1014

Stable State

Example Input 2:

[[9,9,7,9,9,9],[7,7,6,8,9,9],[5,6,5,6,8,8],[1,5,1,4,1,1],[2,1,4,1,1,1],[1,4,1,3,1,1],[1,1,2,1,3,1],[1,2,1,1,1,3]]

Example Output 2:

[[0,0,0,0,0,0],[0,0,0,0,0,0],[0,0,0,0,0,0],[0,0,0,0,0,0],[0,0,0,0,0,0],[0,0,0,0,0,0],[0,0,0,0,0,0],[9,9,0,0,9,9]]

Important Notes

This is a homework for each **team**. Please submit your homework **by one team member. You are asked to write comments to describe the meaning of each part in hw3_p1.py, hw3_p2.py, and hw3_p3.py.**

How to Submit Your Homework? [Both (1) and (2) need to be done!]

(1) Submission in NCKU Moodle

Before submitting your homework, please zip the files (**hw3_p1.docx**, **hw3_p2.py**, and **hw3_p3.py**) in a zip file, and name the file as "學號 1_學號 2_hw3.zip". For example, if your 學號 of your team are H12345678 and H87654321, then your file name is:

"H12345678_H87654321_hw**3**.zip" or "H87654321_H12345678_hw**3**.rar"

When you zip your files, please follow the instructions provided by TA's slides to submit your file using NCKU Moodle platform http://moodle.ncku.edu.tw .

(2) Print out Your Codes and Files

Please print out your files (**hw3_p1.docx**, **hw3_p2.py**, and **hw3_p3.py**) using A4 papers, clearly highlight which papers belong to which problems, and staple and nail these papers together. Then **bring your printed-and-nailed papers to the class on 11/28**. Please make sure your printed version is exactly the same as the submitted version in Moodle.

Have Questions about This Homework?

Please feel free to visit TAs, and ask/discuss any questions in their office hours. We will be more than happy to help you.