CIS368 HW1 (2 programming subtasks) Due: 01/27/2020 11:59 pm

Textbook Exercise 8.12 (Page 311. Financial Application: compute tax) Use array to write a program to compute tax. For each filing status (Single filer, Married jointly or qualifying widow(er), Married filing separately, head of household), there are six tax rates. Each rate is applied to a certain amount of taxable income. For example, from taxable income of $400,000 for a single filer, $8,350 is taxed at 10%,
(33,950 - 8,350) at 15%,
(82,250 - 33,950) at 25%,
(171,550 - 82,550) at 28%,
(372,550 - 171550) at 33%, and
(400,000 - 372,950) at 35%.
The six rates are the same for all filing status, which can be represented in the following array:

```
Double[] rates = {0.1, 0.15, 0.25, 0.28, 0.33, 0.35}
```

The brackets for each rate for all the filing statues can be represented in a two-dimensional array as follows:

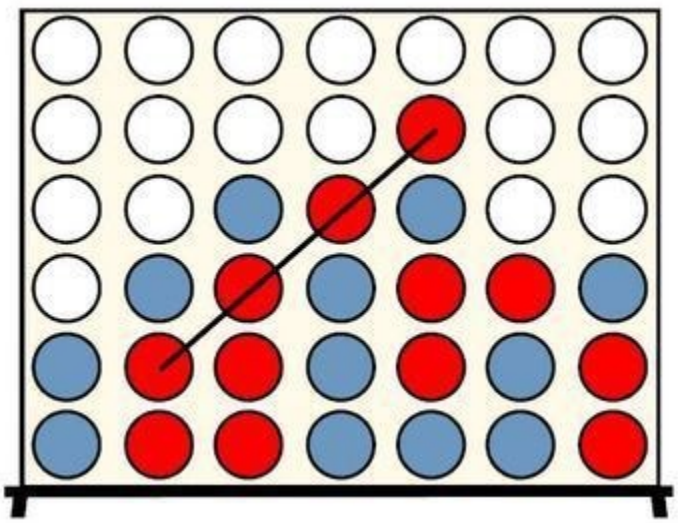
```
int[][] brackets =
      {{8350, 33950, 82250, 171550, 372950},  // Single filer
       {16700, 67900, 137050, 20885, 372950}, // Married jointly
                                               // -or qualifying widow(er)
       {8350, 33950, 68525, 104425, 186475}, // Married filing separately
       {11950, 45500, 117450, 190200, 372950}  // Head of the household
      }
```

Suppose the taxable income is $400,000 for single filers. The tax can be computed as follows:

```
Tax = brackets[0][0] * rates[0] +
          (brackets[0][1] - brackets[0][0]) * rates[1] +
          (brackets[0][2] - brackets[0][1]) * rates[2] +
          (brackets[0][3] - brackets[0][2]) * rates[3] +
          (brackets[0][4] - brackets[0][3]) * rates[4] +
          (400000 - brackets[0][4]) * rates[5]
```

Textbook Exercise 8.20 (page 315)

(Game: connect four) Connect Four is a two-player board game in which the players alternated drop colored disks into a seven-column, six-row vertically suspended grid, as shown below.



The objective of the game is to connect four same-colored disks in a row, a column, or a diagonal before your opponent can do likewise. The program prompts two players to drop a red or yellow disk alternatively. In the preceding figure, the red disk is shown in a dark color and the yellow in a light color. Whenever a disk is dropped, the program redisplays the board on the console and determines the status of the game (win, draw, or continue). Here is a sample run.

```
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
```

```
Drop a red disk at column (0-6): 0
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|R |  |  |  |  |  |  |
```

```
Drop a yellow disk at column (0-6): 3
|  |  |  |  |  |  |  |
```

```
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|R |  |  |Y |  |  |  |
```

. . .

. . .

. . .

Drop a yellow disk at column (0-6): 6

```
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  | R|  |  |  |
|  |  |  | Y| R| Y|  |
|  |  |R | Y| Y| Y|Y |
|R |Y |R |Y | R|R |R |
```

Yellow Wins. Game End!

Frequently asked questions -

(1) Question: Can I use JavaFX making a much prettier user interface instead of using console input? Answer: Yes.

(2) Question: Can I write a web application using HTML and Java Servlet with a pretty user interface? Answer: Yes.