This assignment is designed to give you some practice and experience writing Java applications using arrays. Each question specifies the name that each application should have. Please follow this naming convention. When you have completed the assignment compress all of your files into a single archive using Windows (Right Click folder -> Send To -> Compressed Folder) or OS X (Right Click folder -> Compress). Using a 3rd party compression utility such as WinRAR or 7zip may render your files unreadable and un-markable. Submit single compressed file to D2L. You can resubmit your files as many times as you would like up to the due date and time.

Be sure to include your name and a brief description of your program (as comments) at the top of each file. Pay attention to using good variable names. If you have any questions please check or post to the forum.

If you work in partners, submit only one solution and make sure that both partners' names are in ALL files. If only one name appears, only one person will get the grade. No exceptions.

Submit solutions to question ****2 and 3**** only. It is assumed that you are doing all of the questions and some of the solutions may be referenced in future assignments.

Question 1. (Arrays.java) Write a program that prompts the user to enter in an integer number representing the number of elements (between 2 and 12, use a while loop for input validation) in an integer array. Create the appropriately-sized array and prompt the user to enter in values for each element using a for loop. When the array is full, display the following:

- The values from the array on a single line (comma separated).
- The values from the array on a single line (comma separated) in reverse order.
- The values from the array that have odd numbered values.
- The values from the array that have odd numbered indexes. Do not use an if statement here.
- The maximum from the array and the position (index) where it occurs.

Use a separate for loop for each task. All of this work can be done in a single `main( )` method.

Sample output will be posted to D2L.

(Badge bonus – "High Frequency") Determine the most frequently (absolute and percentage) occurring value in the array. If multiple values are tied then output the last one computed. Eg. output:

```
7 occurs 5 times (50%) in the array.
```

Question 2. (`ArrayLists.java`) Repeat all of the functionality of Question 1 (minus the bonus) but this time use ArrayLists instead of Arrays. Also, add a prompt to allow the user to specify a value. Print the number of occurrences of that value in the list. If the value does not appear, print 0.

Sample output will be posted to D2L.

Question 3. (`Arrays2D.java`) Write a Java program that will show off 2D arrays. Your program should use 2D arrays and nested for loops to reproduce the following output. Note: the array is always 4 x 5 and is filled with random numbers between 1 and 10. Only the count occurrences number is collected from the user. You may do all of this work in a single `main()` method.

```
Creating array (4x5, random numbers)
1     9     5     7     6
2     5     2     1     3
4     8     5     1     2
7     9     1     2     6
Sum of array is: 86
Avg of array is: 4.3
Sum of rows is:
0:28
1:13
2:20
3:25
Max of columns is:
0     1     2     3     4
7     9     5     7     6
Count occurrences of: 6
2
```

Question 4. Write a class that has the following methods. Use nested for loops for all of these array operations:

`public static void printArray(int[][] a)` : print the values stored in the array

`public static int sum(int[][] a)` : compute and return the sum of the values in the array

`public static int[][] sum(int[][] a, int[][] b)` : compute the sum of two arrays and return a new array with those values. Remember, each value in array `a` gets added to the corresponding value in array `b`. Assume that the arrays are the same size.

`public static boolean countOccurences(int[][], int n)` : return `true` if the value n is found in the array. Return `false` if it is not found.

In your `main()` method prompt the user to enter two integers that will be the rows and columns for two 2D arrays. Create the arrays according to the input dimensions and fill them with random values from 0 to 15. Then prompt for a third integer that will be used to test your `isFound()` method. Remember, to do all input and output in your `main()` method. Pass the input to your methods and collect the results using assignment statements. You don't need to do any input checking. Assume the user enters appropriate values. I suggest your write and test your methods one at a time. You can get part marks for the methods that work.

(MINUS 5 marks) if you do the work all the work but in a single main method.

```
java Arrays2D
Enter two integers for rows and columns of the arrays:3 4

First array:
5       2       13      3
9       10      11      7
7       11      1       14

Second array:
7       15      2       1
1       2       2       9
11      2       5       4

Sum first array: 93
Sum second array: 61
Sum of first array + second array:
12      17      15      4
10      12      13      16
18      13      6       18

Enter an integer to search for: 3
3 found in first array?: true
3 found in second array?: false


java Arrays2D
Enter two integers for rows and columns of the arrays:2 6

First array:
3       9       10      10      0       4
4       6       3       3       15      5

Second array:
5       0       0       10      2       0
0       5       13      14      6       2

Sum first array: 72
Sum second array: 57
Sum of first array + second array:
8       9       10      20      2       4
4       11      16      17      21      7

Enter an integer to search for: 0
0 found in first array?: true
0 found in second array?: true
```

Question 5. (`TicTacToe.java`) Write a text-based, 2-person, tic-tac-toe game. This game will combine most of what you have learned so far. You'll use methods and a 2-D array in this game. A general description of how to proceed and what is expected follows below. The flow of your game should be roughly as follows:

Display the instructions for the game
Assume X always goes first
Create the empty board (initialize your 2D array)
Display the board
While there isn't a winner or a tie
    Get the player's move (collect a row and a column)
    While the player's move is not legal
        Get the player's move
    Update the board with the move
    Display the updated board
    Switch turns
Congratulate the winner or declare a tie

The main method declares the 2D array, keeps track of whose turn it is, takes care of collecting the player's move (a row and a column) and executes the flow of the program as described above by calling the necessary methods.

Here are the methods that I will be looking for in your program:

`public static void displayInstructions( )`: This is a void method that takes no arguments. It's job is simply to display the instructions for the game (see below).

`public char[][] createBoard( )`: This method takes no arguments but returns a 3x3 array of characters with each character set to ' ' (space).

`public static void displayBoard(char[][] board)`: This method takes the current game board as an argument and displays it to the screen.

`public static boolean isWinner(char[][] board)`: This method takes the current game board as an argument and determines if there is a winner. If there is, it returns true else return false.

`public static boolean isTie(char[][] board)`: This method takes the current game board as an argument and determines if there is a tie. If there is, it returns true else return false.

`public static boolean legalMove(int row, int column, char[][] board)`: This method checks to see if the user's move is legal. A legal move is inside the board to a space that isn't currently occupied. It returns true or false accordingly.

`public static void updateBoard(int row, int column, char player, char[][] board)`: This method updates the board with the player's move.

Sample output from two separate games is shown below.

```
Instructions:  Fill in a complete row, column or
diagonal with either X's or O's to win
Recall that array indices start at 0!


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


Player X--> Enter a row and column: 1
1

  |   |
----------
  | X |
----------
  |   |
Player O--> Enter a row and column: 0 0

O |   |
----------
  | X |
----------
  |   |
Player X--> Enter a row and column: 1 1
Player X--> Enter a row and column: -1 -1
Player X--> Enter a row and column: 1 5
Player X--> Enter a row and column: 0 2

O |   | X
----------
  | X |
----------
  |   |
Player O--> Enter a row and column: 2 0

O |   | X
----------
  | X |
----------
O |   |
Player X--> Enter a row and column: 1 0

O |   | X
----------
X | X |
----------
O |   |
Player O--> Enter a row and column: 2 2

O |   | X
----------
X | X |
----------
O |   | O
Player X--> Enter a row and column: 1 2

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

Game over!
Congratulations player X, you won!!!
```

```
Instructions:  Fill in a complete row, column or
diagonal with either X's or O's to win
Recall that array indices start at 0!


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


Player X--> Enter a row and column: 1 1

  |   |
----------
  | X |
----------
  |   |
Player O--> Enter a row and column: 0 0

O |   |
----------
  | X |
----------
  |   |
Player X--> Enter a row and column: 2 2

O |   |
----------
  | X |
----------
  |   | X
Player O--> Enter a row and column: 0 2

O |   | O
----------
  | X |
----------
  |   | X
Player X--> Enter a row and column: 0 1

O | X | O
----------
  | X |
----------
  |   | X
Player O--> Enter a row and column: 2 1

O | X | O
----------
  | X |
----------
  | O | X
Player X--> Enter a row and column: 1 0

O | X | O
----------
X | X |
----------
  | O | X
Player O--> Enter a row and column: 1 2

O | X | O
----------
X | X | O
----------
  | O | X
Player X--> Enter a row and column: 2 0

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

Game over!
There was a tie
```