Lab 6

Task 1 Ask the user to first enter the number n. Then take n numbers from the user and print out the average, highest, lowest for these n numbers**. Use an ArrayList to do this.**

Sample output:

```
Enter number of items: 4
Enter 4 numbers: 7
6
3
4
Mean: 5.0
Highest: 7
Lowest: 3
```

Task 2 Given two vectors of length N that are represented with **ArrayLists**, write a code fragment that computes the Euclidean distance between them (the square root of the sums of the squares of the differences between corresponding entries).
Sample run:

```
Enter the size of the vectors: 3
Enter 3 coefficients of the first vector: 1
2
3
Enter 3 coefficients of the second vector: 1
2
1
The Euclidean distance is: 2
```

Task 3 Redo Task 2 using a method. The method should take two **ArrayLists** and should return the Euclidean distance.

Task 4 Write a method to take the transpose of a n x m matrix. The *transpose* of matrix is simply a flipped version of the original matrix. We can transpose a matrix by switching its rows with its columns. For example, if

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

then the transpose of A, denoted by A^T is:

$$A^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}.$$

Read more: https://mathinsight.org/matrix_transpose

Your method should one 2d array and return a 2d array.

Test your method in the main method. Write code that calls your method with the three different inputs below, and ensure the output is what you expect.

$$\begin{bmatrix} 1 & 2 \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

Task 5 Minesweeper-like Write a program that takes 2 arguments M, N and produces an M-by-N array where each entry is randomly occupied or not. In the Minesweeper game, occupied cells represent bombs and empty cells represent safe cells. Print out the array using an asterisk for bombs and a period for safe cells. Then, build and print a new array, where bombs are still represented with asterisks, while safe cells are filled in with the number of neighboring bombs (above, below, left, right, or diagonal).

```
Enter M: 3
Enter N: 5
Matrix:
* * . . .
. . . . .
. * . . .

Solution:
* * 1 0 0
3 3 2 0 0
1 * 1 0 0
```