MATLAB Project: Reduced Echelon Form and ref Name_____

Purpose: To calculate the reduced echelon form by hand and with ref, and to see some effects of roundoff error.
Prerequisite: Section 1.2
MATLAB built-in functions used: `-` , `/`
M-files used: `replace`, `swap`, and `scale` from the `laydata5` toolbox or from pearsonhighered.com/lay.

Background: Read about elementary row operations and reduced echelon form in Section 1.2. To learn about the functions from Laydata5 Toolbox, use **`help`** or see Lay's Study Guide.

1. Type **`rowdat`** to get the two matrices A and B below. For each of them, first calculate the reduced echelon form by hand, and then use the function **`ref`** from the Laydata5 Toolbox to calculate the reduced echelon form again.

(a) (Hand) The matrix A below is Example 2 in Section 1.2, where an echelon form is calculated. Repeat here the row operations done in the text, and then finish calculating by hand its reduced echelon form. Show all steps:

$$A = \begin{bmatrix} 0 & -3 & -6 & 4 & 9 \\ -1 & -2 & -1 & 3 & 1 \\ -2 & -3 & 0 & 3 & -1 \\ 1 & 4 & 5 & -9 & -7 \end{bmatrix} \sim$$

(b) (MATLAB) Type **`format rat`** and then **`ref(A)`**. Is the output identical to what you obtained above? _____ (If not, redo hand calculations.)

Remarks: MATLAB has a built-in command `rref` for finding the row reduced echelon form of a matrix. In general, Laydata5 Toolbox command `ref` is faster because it does not check for rational entries as `rref` does. Except for this rational number issue, the code for `ref` is the same as `rref`.

2. Let $B = \begin{bmatrix} -0.1 & 0.1 & 2 \\ 0.3 & 0.2 & 0.7 \\ 0 & 0.5 & 6.7 \end{bmatrix}$.

(a) (hand) Calculate the reduced echelon form of B. Please show all steps. *Hint:* do all scaling at the end.

(b) (MATLAB) Type **`ref(B)`**. Is the output identical to what you obtained above? _____
(If not, redo hand calculations.)

Remarks For the majority of small matrices like those used in linear algebra courses, `ref` will return a very accurate result as it does in the two examples above. One of the reasons `ref` is accurate is that it does not pivot on a position where the value is extremely small. Such a number is often inaccurate in many digits as a result of roundoff error during previous row operations, and it is even possible that theoretically it ought to be a true zero. The usual algorithm for Gaussian elimination chooses the next pivot by looking "to the right and down" for the first nonzero entry in the first nonzero column. If that entry happens to be a very inaccurate number, pivoting on it can lead to a very wrong final result.

Thus it is wise when doing row reduction with a computer or calculator to check the size of potential pivots and not to use one that is extremely small. (The question of what is "extremely small" is a matter of judgment and depends largely on how many digits your computer arithmetic keeps.) The functions `ref` and `rref` have a tolerance variable called `tol` for this purpose. A user can specify a value for `tol`, but if a value is not specified, then `ref` uses a default value. If the absolute value of a number is less than `tol`, then `ref` will not pivot on that position.

3. (MATLAB) Use the same matrix *B* as in question 2. Here you will force `ref(B)` to return the wrong answer by making the value of `tol` too small, and you will experiment to figure out a good estimate for what is the default value of `tol`.

(a) Type each of the following commands and record the result:

```
ref(B, 1e-15)
```

```
ref(B, 1e-16)
```

Now you can be certain that the default value for `tol` is smaller than 10^{-15} but not smaller than 10^{-16}. Explain why:

(b) Experiment to find more precise bounds for the default value of `tol` and record the ones you find:

Hint: Type **`ref(B,9e-16)`**, then **`ref(B,1e-15)`**. Is `tol` between `9e-16` and `1e-15`? Etc.