CS 1112 Spring 2020 Test 2A April 21 4:30 PM EDT + 48 hr

Name: _____ NetID: _____
(Legibly print last name, first name, middle name)

Statement of integrity: *I did not, and will not, violate the rules of academic integrity on this exam.* _____
(Signature)

Instructions:

- This is an open-note exam and is designed to take approximately 50 minutes to complete. No collaboration is allowed.
- The exam is worth a total of 50 points.
- Read each problem completely before starting it.
- If a question is unclear, e-mail Dr. Muhlberger; do not ask anyone else. Check Canvas before submitting your exam in case we need to announce any clarifications to the whole class.
- Clarity, conciseness, and good programming style count for credit.
- Indicate your final answer. If you supply multiple answers, you may receive a *zero*.
- Use only MATLAB code. No credit for code written in other programming languages.
- Assume there will be no input errors.
- Vectorized code is not required (but it may make things easier to write).
- You may write a subfunction as part of your solution if you think it will clarify your code.
- Do not use `switch`, `try`, `catch`, `break`, `continue`, or `return` statements.
- **Do not use built-in functions that have not been discussed in the course.** Limit yourself to the following MATLAB predefined functions: `abs`, `sqrt`, `rem`, `floor`, `ceil`, `round`, `min`, `max`, `sum`, `rand`, `zeros`, `ones`, `linspace`, `length`, `size`, `uint8`, `double`, `transpose`, `strcmp`, `input`, `fprintf`, `disp`

Examples:
`rem(5,2)` → 1, the remainder of 5 divided by 2
`min(-4,3)` → -4, smallest argument
`max(-4,3)` → 3, largest argument
`sum([0 4; 1 -1])` → [1 3], vector of column sums of the *matrix* argument
`rand()` → a random real value in the interval (0,1)
`zeros(1,4)` → 1 row 4 columns of zeros
`linspace(3,5,10)` → a vector of 10 real numbers evenly distributed in the interval [3,5]
`length([2 4 8])` → 3, length of a vector
`[nr,nc,np]=size(M)` → dimensions of `M`: `nr` rows, `nc` columns, `np` layers
`uint8(4.7)` → the integer (type `uint8`) value 5
`transpose(A)` → `A'`, matrix `A` with rows and columns switched
`strcmp('cat','Cat')` → 0, the two strings are not identical
`strcmp('dog',transpose(['d'; 'o'; 'g']))` → 1: to compare a row vector to a column vector using `strcmp`, take a transpose

Question 1. (14 points)

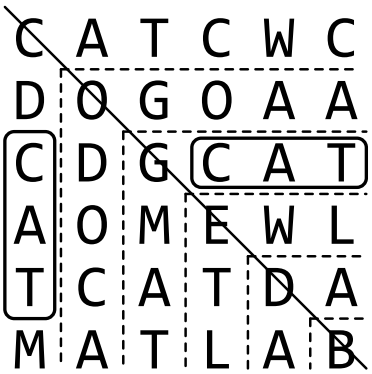
Write a function `extractLastName` to extract a person's last name from a character vector having the following format: names appear in the order *first name, middle name(s), last name* and are separated from each other by one or more plus characters ('+'). Additionally, there may be extra plus characters after the last name.

The function `extractLastName` should accept a single input argument: a character vector containing the person's full name in the above format. It should return a single output: a character vector containing only the last name that appears in the input. Regardless of how many "names" appear in the input, whichever is last (farthest to the right) should be interpreted as the last name, even if it's the only name. If no name appears (i.e., the input is empty or all plus signs), then the returned character vector should be empty.

For example, if the input is '`Geralt+of+Rivia++`', then the returned value should be '`Rivia`'.

Question 2. (20 points)

Write a function `countWords` to search for a word in a square character matrix and count the number of times it occurs, subject to a few constraints: Words may be found horizontally (left-to-right) above the main diagonal of the matrix, or vertically (top-to-bottom) below the main diagonal of the matrix. Words may not use any letters on the main diagonal, nor may they be "backwards" (right-to-left or bottom-to-top).



The function `countWords` should take two input arguments: the first is the matrix to search in, and the second is the word to search for (a character row vector). It should return one output: the number of times that the word can be found.

Example: In this 6x6 character matrix, the word 'CAT' occurs twice according to the above rules, so the function should return 2. (It also occurs 3 other times in ways that violate the rules; these are not counted.)

Question 3. (16 points)

Implement the following technique for making images look like old video. Every odd row of the image should be blurred horizontally, while every even row of the image should be set to black.

To blur the odd rows, each color intensity (red, green, and blue) of each pixel should be replaced by the average of itself and that of its left and right neighbors, 2 on each side (so nominally 5 pixels participate in the average). If a pixel doesn't have 2 neighbors on one side, use the average of itself and the neighbors it does have.

Name your function `makeLoFi`. It should accept one input argument, a 3-dimensional `uint8` array representing the original color image, and it should return one output value, a 3-dimensional `uint8` array of the same size representing the processed image.