Adhere to the Code of Academic Integrity. You may discuss background issues and general strategies with others and seek help from course staff, but the implementations that you submit must be your own. In particular, you may discuss general ideas with others but you may not work out the detailed solutions with others. It is never OK for you to see or hear another student’s code and it is never OK to copy code from published/Internet sources. If you feel that you cannot complete the assignment on your own, seek help from the course staff.

When submitting your assignment, follow the instructions summarized in Section 4 of this document.

Do not use the break or continue statement in any homework or test in CS1132.

Completing this assignment will help you solidify your understanding of manipulating 2-d arrays, extracting sub-arrays from a 2-d array, using vectorized code, and writing and calling subfunctions. This assignment will also give you the opportunity to practice plotting data with Matlab.

1 Summarizing Visit Statistics at a Veterinary Clinic

A local veterinary clinic keeps a record book of the animals that have visited for a check-up and takes note of the problem(s) that each animal suffers from. This record book has 11 columns where the first column indicates the type of animal while each of the 10 remaining columns refers to a body part that the animal has problems with. The manager of the clinic hopes to obtain some statistics about which body part plagues which animal the most, in an effort to recruit more specialized veterinarians.

1.1 Implement the following function as specified.

```matlab
function stats = computeVetStats(record)
% This function summarizes and plots statistics based on records taken at a
% local veterinary clinic. record is a 2D array with n rows and 11 columns
% where n is the number of animals that have visited the clinic thus far.
% The values of the first column of record represent the type of animal:
% 1-dog, 2-cat, 3-other animal. Assume that the clinic has dealt with
% each type of pet at least once. Columns 2 to 11 of record classify the
% problem that the animal suffers from. An animal may suffer from more than
% 1 problem simultaneously. This is indicated by record(i,j) = 1 if animal
% i suffers from the problem corresponding to column j for j>=2 and 0
% otherwise. The following list classifies the problems that the animal may
% suffer from:
% column 2 - mouth infection
% column 3 - paw problem
% column 5 - tail problem
% column 6 - urinary problem
% column 7 - gastrointestinal problem
% column 8 - leg problem
% column 9 - ear infection
% column 10 - breathing problem
% column 11 - heart problem
% This function returns stats which is a 3x10 2-d array where stats(i,j)
% is the fraction of type i animals that have type j incidents. (E.g.,
% stats(2,1) is the fraction of cats seen at the clinic that have mouth
% infection. This function also plots in one figure the frequencies of
% incidents for each animal type, i.e., plot each row of stats as one
% curve in the figure (so the plot has three curves): blue for dogs, green
% for cats, and red for other animals.
```
Your figure must include an appropriate title and appropriate x-axis and y-axis labels. The x-axis shows
the incident type numbers, not the incident names. As we are plotting the statistics for three different
categories of animals in one figure, a different color should be used for each animal as indicated in the
function description. In addition, in order to easily interpret the results, having a legend is useful to
label each solid line. A legend can be easily incorporated using the `legend` command. For example, if
the figure includes two curves the function call could be `legend('First curve description', 'Second
curve description')`.

See Section 2.11 Introduction to Plotting in the textbook for more example code on plotting. Matlab’s
documentation is also a recommended resource to learn more about customizing your plots.

1.2 Test your function by writing a script file `assignment1b.m` in which you create at least one set of test
data and call your function using the test data. A set of test data is an `n`-by-11 matrix where `n` is greater
than 3 and all three animal types are represented in the matrix. Make sure that you check at least these
scenarios:

- an animal seen suffers from exactly one problem type
- an animal seen has multiple problems
- an animal seen has no problems
- a problem type is not observed in any of the seen animals

Organize and comment your code in `assignment1b` to clearly indicate what scenarios are being tested.

2 Sudoku Puzzle

A completed Sudoku puzzle is a 9 × 9 grid such that:

- each row,
- each column,
- and each of the nine 3 × 3 subgrids (defined by the interior bold grid lines in Figure 1)

contains the numbers 1-9 only once. An example of a completed Sudoku puzzle is shown in Figure 1.

![Figure 1: An example of a completed Sudoku puzzle.](image)

Your task is to write a function that determines whether a user’s attempt at completing the Sudoku puzzle
is successful, as specified below:
function isCompleted = isSudoku(A)
    % This function checks if A is a completed Sudoku puzzle. A is a 9-by-9
    % array that represents a user’s attempt at Sudoku. The function sets
    % isCompleted to 1 if A is a successfully completed Sudoku puzzle and 0
    % otherwise.
    %
    % A Sudoku puzzle is completed if each row, each column, and each of the 9
    % 3x3 subgrids, contains the numbers 1-9 (only once). An example of a
    % completed Sudoku puzzle is shown below.
    %
    % A = [4 3 5 2 6 9 7 8 1; ...
    %     6 8 2 5 7 1 4 9 3; ...
    %     1 9 7 8 3 4 5 6 2; ...
    %     8 2 6 1 9 5 3 4 7; ...
    %     3 7 4 6 8 2 9 1 5; ...
    %     9 5 1 7 4 3 6 2 8; ...
    %     5 1 9 3 2 6 8 7 4; ...
    %     2 4 8 9 5 7 1 3 6; ...
    %     7 6 3 4 1 8 2 5 9]

    In order to complete this function, you must create and make use of at least one subfunction. Recall
    that subfunctions are useful especially if there are tasks in the main function that need to be done repeatedly.
    Design your subfunctions to be meaningful, not trivial, for your solution of this problem. For example,
    a trivial use of subfunctions would be to have your subfunctions perform only very basic mathematical
    operations or involve only a very simple if-statement.
    In addition, try to make your code as efficient as possible. That is, once your code determines that a single
    row, column, or 3 \times 3 block violates the conditions for a Sudoku puzzle, there is no need to check whether
    the remaining rows, columns, or 3 \times 3 blocks satisfy the specified criteria.
    The built-in functions sort and reshape may be useful but are not essential. Search the Matlab documen-
    tation to learn how to use these functions if you are interested.

3 Self-check list

The following is a list of the minimum necessary criteria that your assignment must meet in order to be
considered satisfactory. Failure to satisfy any of these conditions will result in an immediate request to
resubmit your assignment. Save yourself and the graders time and effort by going over it before submitting
your assignment for the first time.

Note that, although all of these are necessary, meeting all of them might still not be sufficient to consider
your submission satisfactory. We cannot list everything that could be possibly wrong with any particular
assignment!

Δ Comment your code! If any of your functions is not properly commented, regarding function purpose
and input/output arguments, you will be asked to resubmit.

Δ Suppress all unnecessary output by placing semicolons (;) appropriately. At the same time, make sure
that all output that your program intentionally produces is formatted in a user-friendly way.

Δ Make sure your functions’ names are exactly the ones we have specified, including case.

Δ Check that the number and order of input and output arguments for each of the functions matches
exactly the specifications we have given.

Δ Test each one of your functions independently, whenever possible, or write short scripts to test them.

Δ Check that your scripts do not crash (i.e., end unexpectedly with an error message) or run into infinite
loops. Check this by running each script several times in a row. Before each test run, you should type
the commands `clear all; close all;` to delete all variables in the workspace and close all figure windows.

4 Submission instructions

1. Upload files `computeVetStats.m`, `assignment1b.m`, and `isSudoku.m` to CMS in the submission area corresponding to Assignment 1b before the deadline.

2. When the scores are released, read the grader’s feedback carefully.

3. If you need to resubmit, fix all the problems and go back to Step 1! Otherwise you are done with this assignment. Well done!