Question 1. (15 points)

(a) Assume that the following script has just been executed by MATLAB:

```
v = [212, 192];

w = uint8(v);

a = v(1) + v(2);

b = w(2) - w(1);

c = 2.0 * w(1);

d = (v(1) + w(2)) / 4;

e = v(1) > 200;
```

What are the values and types of the variables a through e, as would be seen in the MATLAB Workspace? Valid types are: `double`, `uint8`, `logical`, `char`, `cell`.

Variable	Value	Type
a	404	`double`
b	0	`uint8`
c	255	`uint8`
d	64	`uint8`
e	1 (true)	`logical`

(b) Fill in the value of the following `size()` expressions (your answer should be a short vector of integers):

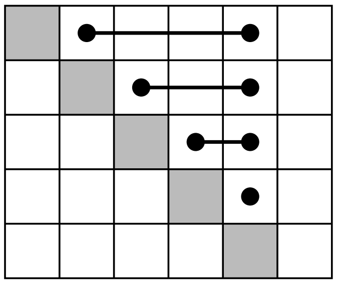
```
a= zeros(3, 5);
k= 1;

adims= size(a(2:3, k:k+2))       % _______ [2, 3] _______


b= zeros(240, 360, 3);
m= 96;

bdims= size(b(120, m-3:m+3, :))  % _____ [1, 7, 3] ______
```

Question 2. (10 points)

Fill in the placeholders $\boxed{\texttt{A}}$ through $\boxed{\texttt{F}}$ with variable names and expressions so that the code below performs a *row-major* traversal of the elements of matrix `M` above and to the right of (but excluding) the diagonal and excluding the last column, as illustrated below (do not make any assumptions about the size of `M` based on the illustration).

Hint: Write out the indices of the first few elements that are printed, in order, then look for a pattern. Consider matrices taller or wider than the illustration, and check edge cases.

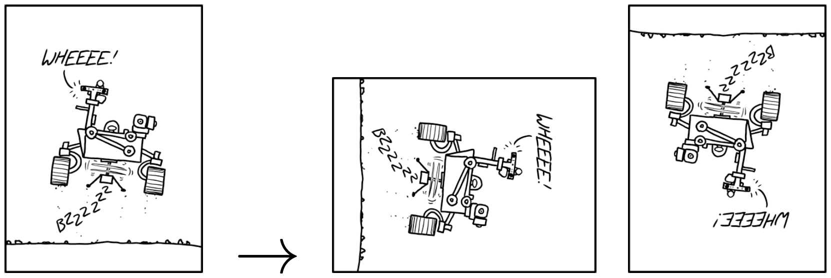
```
[nr, nc] = size(M);

for [A] = [B]:[C]

    for [D] = [E]:[F]

        disp(M(r,c))

    end

end
```



- A: r
- B: 1
- C: nr
- D: c
- E: r+1
- F: nc-1

Question 3. (20 points)

Declare and implement a function, `rotateImage`, to rotate an image clockwise by a multiple of 90°. The function should have two input arguments: the image to rotate (a 3D `uint8` array) and the number of rotations to perform (an integer ≥ 0); it should return the rotated image (also a 3D `uint8` array). For maximum points, make effective use of vectorized code and avoid redundant work. You do not need to write a specification comment.



input[1] one rotation two rotations

Hint: Draw a picture of the input image, labeling the indices of a few pixels. Then draw the output after a single rotation and label the indices where those pixels ended up. Write a scalar formula for copying a single pixel before attempting a vectorized solution (a correct loop-based solution is better than an incorrect vectorized one).

```
function out = rotateImage(img, nturns)

% Four rotations is the same as doing nothing
nturns = rem(nturns, 4);
out = img;  % Handles case when nturns is 0
for k = 1:nturns
    [nr,nc,np] = size(img);
    % Reset output to rotated size
    out = uint8(zeros(nc,nr,np));
    for r = 1:nr  % Loop over rows of input image
        out(:,nr-r+1,:) = img(r,:,:);
    end
    % Copy output to input for use in next iteration
    img = out;
end
```

Alternative rotations (not an exhaustive list):

```
    for c = 1:nc  % Loop over columns of input image
        out(c,:,:) = img(nr:-1:1,c,:);  % Reverse rows on RHS
    end
```

```
    for p = 1:np  % Loop over layers
        out(:,nr:-1:1,p) = img(:,:,p)';  % Transpose operator
    end
```

[1]Image from xkcd.com

Question 4. (25 points)

A monkey at a typewriter has filled a page with text, stored in a character matrix. We want to look for broken fragments of Shakespeare in this document. We proceed by *decomposing the problem*.

(a) The first step is to search for a single word in a single line (row) of text. **Implement the following function** to perform this search starting from a given column:

```
function c = findInLine(w, txt, c)
% Search for string `w` within character row vector `txt`, starting from
% column `c`.  Return the column on which the first letter of the word was
% found, or 0 if the word is not present.
% Example: findInLine('be', 'toxbe', 3) is 4, but
%          findInLine('to', 'toxbe', 3) is 0 (because 'to' is not found at
%             or after column 3)
```

```
n = length(w);
nc = length(txt);
while c <= nc - n + 1 && ~strcmp(w, txt(c:c+n-1))
    c = c + 1;
end
if c > nc - n + 1
    c = 0;
end
```

(b) Assume that a collaborator has completed the next step: searching for a single word on a page of text, starting from a given position. They have written function `findWord()` (using your `findInLine()` as a helper—thanks!) as specified below:

```
function [r, c] = findWord(w, txt, r0, c0)
% Search for word `w` in lines of character matrix `txt`, starting from
% column `c0` of row `r0`.  Return the row `r` on which the word is found,
% and the column `c` containing its first letter, or return (r=0,c=0) if
% the word is not present.
% Edge cases: If c0 > # cols of txt, then search begins on the next line,
% at index (r+1,1).  If r0 is 0, return (0,0) immediately.
```

The final step is to search for all words of a phrase, in order (but possibly separated), on the page. **Implement function `findPhrase()`**, specified below. Make effective use of (but do not implement!) function `findWord()`.

```
function f = findPhrase(p, txt)
% Search for each word of `p` (a 1D cell array of strings) in lines of 2D
% character matrix `txt`, requiring that each word occurs somewhere after
% the previous one (either later in the same row, or on a subsequent row;
% found words may not overlap).  Return true if all words are found in
% order, false otherwise.
% Example: findPhrase({'to','be','or'}, ['ztoq';'beor']) is true,
%          findPhrase({'muse','of','fire'}, ['bmuseo';'fireof']) is false.
```

```
r = 1;
c = 1;
k = 1;
while r > 0 && k <= length(p)
    [r,c] = findWord(p{k}, txt, r, c);
    % Start next search after end of previous word
    c = c + length(p{k});
    k = k + 1;
end
f = r > 0;
```

A `for`-loop, though less efficient, could be used instead, since `findWord()` will "latch" a "not found" result.

Question 5. (30 points) Consider student vaccination data stored in a file like the following:

```
2021-03-01 M 3141592 Cody Clay
2021-03-02 P 2718282 Mae Goodwin
2021-03-15 J 1618034 Hakim Umayya
2021-03-17 P 2718282 Mae Goodwin
```

The records are ordered by the date a vaccine dose was received, which occupies the first 10 columns of each line. Column 12 contains an abbreviation code for the type of vaccine. A student's 7-digit ID number starts at column 14, and their name begins at column 22. A student is "done" with their appointments after one dose of vaccine code 'J', or after two doses of any other vaccine (the same student may appear on two different rows; if they do, their vaccine code will be the same each time).

(a) Implement the following function to extract student IDs, names, and vaccine codes. As an example, the outputs for the above file snippet would look like this:

$$\texttt{ids} = \begin{bmatrix} 3141592 \\ 2718282 \\ 1618034 \\ 2718282 \end{bmatrix} \qquad \texttt{codenames} = \begin{Bmatrix} \texttt{`M'}, & \texttt{`Cody Clay'} \\ \texttt{`P'}, & \texttt{`Mae Goodwin'} \\ \texttt{`J'}, & \texttt{`Hakim Umayya'} \\ \texttt{`P'}, & \texttt{`Mae Goodwin'} \end{Bmatrix}$$

Hint: If you have trouble with the file operations, start by writing code to process a single line of text to produce a single row of output.

```
function [ids, codenames] = readVaxRecords(filename)
% Read and parse student vaccination records from file `filename`.
% The student ID from each line will be stored in the corresponding element
% of `ids` (a numeric array), while the vaccine code and student's name
% are stored in columns 1 & 2 of that row in 2D cell array `codenames`.
```

```
%ids = [];          % Initializing is optional, since we are
%codenames = {};    % assuming at least one line is in the file.

fid = fopen(filename, 'r');
k = 0;
while ~feof(fid)
    k = k + 1;
    s = fgetl(fid);
    ids(k) = str2double(s(14:20));
    codenames{k,1} = s(12);
    codenames{k,2} = s(22:length(s));
end
fclose(fid);
```

(b) To determine which students are "done" with their vaccine sequence, we need a way to determine whether a student (with code other than 'J') appears twice in the dataset. Observe that if records were sorted by ID rather than by date, then students who occur twice will have their ID on adjacent rows.

Write a script to print the names of students who are "done" with their vaccine sequence, based on data in a file named `vaccinations.txt`. Your script should make effective use of `readVaxRecords()` and `sort()`. Do not print a student's name more than once.

Example: For the file on the previous page, this script should output:

```
Hakim Umayya
Mae Goodwin
```

```
[ids, codenames] = readVaxRecords('vaccinations.txt');
[sids, p] = sort(ids);

k = 1;
while k <= length(ids)
    idx = p(k);
    % Note: sids(k) == ids(idx)
    if codenames{idx,1} == 'J'
        disp(codenames{idx,2})
    elseif k < length(ids) && sids(k) == sids(k+1)
        disp(codenames{idx,2})
        k = k + 1;  % Skip next row
    end
    k = k + 1;
end
```