Question 1: (10 points)

(a) What is the output from executing the following fragment? Write the word "error" instead of the output if executing the fragment would cause a run-time error.

```
M = [4 3 2; ...
     3 2 1; ...
     1 2 3];
for k = 1:3
    M(k,k) = M(M(k,k)-1, k);
end
disp(M)
```

Solution:

```
 1     3     2
 3     3     1
 1     2     1
```

(b) Consider the script below which reads a plain text file:

```
% Read a text file

fid = fopen('myfile.txt','r');

m = 0;  % Solution: initialize counter

while ~feof(fid)

    s = fgetl(fid);

    % Solution: increment counter of 1st three letters are 'POS'
    if  length(s)>=3 && strcmp('POS',s(1:3))
       m = m + 1;
    end

end

disp(m)  % Solution: display the no. of lines that start with 'POS'

fclose(fid);
```

Add code to the script to determine and print the number of lines in the file that begins with the string `'POS'`. Do not modify or cross out the given statements and do not use built-in functions `find` or `strfind`.

Question 2: (20 points)

(a) Implement function `findMaxInMatrix` as specified. *Do not* use any built-in functions other than `size`.

```
  function [row,col] = findMaxInMatrix(M)
  % M(row,col) is the largest value in matrix M.  row and col are scalars.
  % Assume M is not empty.  If the max value appears in multiple
  % locations in M, (row,col) may be any one of those locations.
```

Solution:

```
[nr,nc] = size(M);
row= 1;
col= 1;
for r=1:nr
   for c=1:nc
      if M(r,c) > M(row,col)
          row= r;
          col= c;
      end
   end
end
```

(b) Implement function `stringLengths` as specified.

```
function len = stringLengths(S)
% Find the lengths of the strings in S, a 1-d cell array of strings.
% len is a vector that has the same length as cell array S.  len(k) is the
% length of the kth string in S.  S is not empty.
```

Solution:

```
len= zeros(1,length(S));  % initialization is not necessary

for r= 1:length(S)
   len(r) = length(S{r});
end
```

Question 3: (25 points)

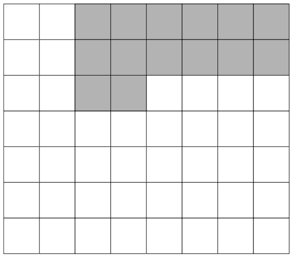
Implement function `hide` as specified. *Do not* use vectorized code. Similar to the digital steganography problem in Project 4, you will modify the 3-d array of a color image by adding to (or subtracting from) the RGB values of a set of pixels.

```
function P = hide(P,vec,cs)
% Hide the values of vector vec in the image data array P, starting at pixel (1,cs).
% P is a 3-d uint8 array from a color image. (type uint8)
% vec is 1-d array of positive integer values; each value is less than 5. (type double)
%   Each value of vec is to be added to (or subtracted from) an RGB value of one pixel,
%   so the length of vec is the number of pixels to modify.  For the kth pixel to be
%   modified, select randomly one layer (red, green, or blue) with equal likelihood and
%   add (or subtract) vec(k).  Choose to add or subtract so that the range of type
%   uint8 is handled correctly.
% cs is a valid column index of array P.  Start the modification at pixel (1,cs),
%   continue to the right, and if necessary continue on the rows below starting at
%   column cs.  (See example diagram at bottom of page.)
% Assume array P is big enough to hide all the values in vec.
```

Solution:

```
n= length(vec);
[nr,nc,np]= size(P);
k= 1;
r= 1;
c= cs;

while k<=n

    % Modify P(r,c) at a randomly chosen layer
    layer= ceil(rand*3);

    if P(r,c,layer)>250  % other thresholds possible
        P(r,c,layer)= P(r,c,layer) - vec(k);
    else
        P(r,c,layer)= P(r,c,layer) + vec(k);
    end

    % update counters
    k= k+1;
    if c<nc
        c= c+1;
    else
        c= cs;
        r= r+1;
    end
end
```

Example: cs=3 and length(vec)=14.
The pixels to be modified are shown in gray.



Question 4: (20 points)

Consider a Student structure (struct) that has three fields: `name`, `netID`, and `major`. All three fields store strings. Assume that each student has only one major and can be in only one college. Implement function `committee` as specified. Your code should be efficient—avoid unnecessary iterations.

```
function committee(E,A)
% Display all possible 3-member committees with one Engineering and two A&S students.
% The three members must have DIFFERENT majors--e.g., an Engineering CS major and an
% A&S CS major cannot both be on a committee.
% E and A are 1-d arrays of Student structs; each array has a length greater than 2.
% Array E holds the data of the Engineering candidates; array A holds the data of the
% A&S candidates.
% For each possible committee, display the NetIDs of the three members on one line.
% Display only unique committees.  For example, the committees "ac14 raf9 jer88" and
% "ac14 jer88 raf9" are duplicates and should not both be displayed.
```

Solution:

```
for i= 1:length(E)

   for j= 1:length(A)-1

       for k= j+1:length(A)

           if ~strcmp(E(i).major, A(j).major) && ...
              ~strcmp(A(j).major, A(k).major) && ...
              ~strcmp(E(i).major, A(k).major)

                disp([E(i).netID ' ' A(j).netID ' ' A(k).netID])
           end
       end
   end
end
```

A more efficient solution:

```
for i= 1:length(E)
   for j= 1:length(A)-1
       if ~strcmp(E(i).major, A(j).major)
          for k= j+1:length(A)
             if ~strcmp(A(j).major, A(k).major) && ...
                ~strcmp(E(i).major, A(k).major)
                disp([E(i).netID ' ' A(j).netID ' ' A(k).netID])
           end
       end
   end
end
```

Question 5: (25 points)

Consider the string `'matt uses matlab'`. We say that the word `'uses'` is located at index 6; i.e., the location of a word in a string is the index of the first letter of the word. Implement function `findPrefix` as specified. Note the example on the bottom of the page. *The only built-in functions allowed are* `size`, `length`, `strcmp`, and `cell`.

```
function C = findPrefix(M,p)
% Locate in matrix M all the words that begin with the string in p.
% M is a 2-d array of characters.
% p is a 1-d array of characters and is the prefix to locate.
% Assume all letters in M and p are in lower case.
% C is a 1-d cell array.  The rth cell in C is a vector, possibly empty, storing
% the locations (column indices) of the prefix p in row r of matrix M.
```

Example: The function call `C = findPrefix(M, 'mat')` where

```
    M= ['there is a mat in the lab'; ...
        'there is a bat in the lab'; ...
        'matt uses matlab on a mat'; ...
        'format a plot in matlab  ']
```

results in a cell array `C` of length 4. The first cell stores the value 12, the second cell stores the empty vector, the third cell stores the vector [1 11 23], and the fourth cell stores the value 18.

Solution:

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

C= cell(nr,1);  % this initialization is not necessary

for r= 1:nr

   % Check row r of matrix M for the prefix
   v= [];

   for c= 1:nc-n+1

       if strcmp(M(r,c:c+n-1),p) && (c==1 || M(r,c-1)==' ')

           v= [v c];
       end
   end

   C{r}= v;
end
```