Question 1: (15 points)

(a) Implement this function:

```
function tot = sumDiag(M)
% tot is the sum of the elements on the main diagonal of numeric square matrix M.
% A matrix is square if its number of rows and number of columns are the same.
% Assume M is not empty.  For example, if M is
%   [ 10   1   2 ; ...
%      0  30  99 ; ...
%     -3   3  20 ]
% then tot is 60.
%
% THE ONLY BUILT-IN FUNCTION ALLOWED IS size.
```

Example solution:

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

tot = 0;

for k= 1:nr

    tot = tot + M(k,k);
end
```

(b) Complete the statement below to assign to variable `ch` a randomly generated capital (upper case) letter; each of the 26 letters in the alphabet should be equally likely to occur. *Only the built-in functions listed on the cover page of this exam are allowed.*

```
ch = _____________________________________________________________________________

% Example solutions:
%
%   char( floor(rand*26)          + 'A' )
%   char( floor(rand*26)          + double('A') )
%   char( floor(rand*('Z'-'A'+1)) + 'A' )
%   char(  ceil(rand*26) - 1      + 'A' )
```

Question 2: (30 points)

Implement this function:

```
function newIm = enlargeImage(Im)
% Perform 2-d interpolation on all three layers of image data Im.
% Im is an nr-by-nc-by-3 array of type uint8 elements.  The interpolated data is
%   added between existing data points so array newIm (type uint8) is
%   (2*nr-1)-by-(2*nc-1)-by-3.
% Use the simple average as the interpolated value (see example below).
% You may use built-in function zeros for initialization but otherwise
%   DO NOT USE VECTORIZED CODE.
```

Example solution:

```
[nr,nc,np] = size(Im);

wideIm = uint8(zeros(nr,2*nc-1,np));
newIm = uint8(zeros(2*nr-1,2*nc-1,np));

% NOTE: above initialization not necessary.  OK to cast original
% matrix Im to a double and then cast newIm as uint8 at the end.

for p = 1:np
    for r = 1:nr
        for c = 1:nc-1

            wideIm(r,2*c-1,p) = Im(r,c,p);

            wideIm(r,2*c,p) = Im(r,c,p)/2 + Im(r,c+1,p)/2;
            % NOTE: if Im is uint8, then
            %     (Im(r,c,p) + Im(r,c+1,p))/2 is incorrect
        end
        wideIm(r,2*nc-1,p)= Im(r,nc,p);
    end
end

for p = 1:np
    for c = 1:2*nc-1
        for r = 1:nr-1

            newIm(2*r-1,c,p) = wideIm(r,c,p);

            newIm(2*r,c,p) = wideIm(r,c,p)/2 + wideIm(r+1,c,p)/2;
        end
        newIm(2*nr-1,c,p) = wideIm(nr,c,p);
    end
end
```

Hint: In 2-d interpolation, work with one dimension at a time. For example, you can first add the interpolated columns and then add the interpolated rows. For example

$$\overset{\textit{One layer of } \texttt{M}}{\begin{bmatrix} 250 & 50 \\ 20 & 100 \\ 10 & 130 \end{bmatrix}} \rightarrow \overset{\textit{Interpolate columns}}{\begin{bmatrix} 250 & 150 & 50 \\ 20 & 60 & 100 \\ 10 & 70 & 130 \end{bmatrix}} \rightarrow \overset{\textit{Interpolate rows}}{\begin{bmatrix} 250 & 150 & 50 \\ 135 & 105 & 75 \\ 20 & 60 & 100 \\ 15 & 65 & 115 \\ 10 & 70 & 130 \end{bmatrix}}$$

Question 3: (30 points)

(a) Implement this function:

```
function z = overlap(diskA, diskB)
% z is 1 (true) if diskA and diskB overlap; otherwise z is 0 (false).
% diskA and diskB are each a disk structure with the following fields:
%   x: x-coordinate of center of disk
%   y: y-coordinate of center of disk
%   radius: radius of disk
```

Example solution:

```
dis = sqrt((diskA.x - diskB.x)^2 + (diskA.y - diskB.y)^2);

if dis < diskA.radius + diskB.radius

    z = 1;
else
    z = 0;
end
```

(b) Implement the following function to return the indices of disk triplets that overlap. Three disks form a triplet if every disk overlaps with each of the other two. Make effective use of function `overlap` from part (a). Your code should be efficient—avoid unnecessary iterations.

```
function idx = diskTriplets(D)
% D is a 1-d array of disk structures; each structure has fields as defined in
%   part (a). Assume D has a length greater than 3.
% idx is a vector of indices indicating all triplet overlap combinations. For example,
%   if disks 2, 4, and 5 form a triplet and disks 3, 4, and 6 form a triplet, idx
%   should be the vector [2 4 5 3 4 6].  Other orderings of triplets are acceptable,
%   however each triplet should only appear once.
```

Example solution:

```
n = length(D);

idx = [];

for i = 1:n - 2

    for j = i + 1:n - 1

        for k = j + 1:n

            if overlap(D(i), D(j)) && ...
               overlap(D(j), D(k)) && overlap(D(i), D(k))

                idx = [idx, i, j, k];
            end
        end
    end
end
```

Question 4: (25 points)

We will split a string into two parts at the first occurrence of a “marker.” For example, if the original string is ‘acagttaga’ and the marker is ‘ag’, then we split the original string into these two parts: ‘ac’ and ’agttaga’. Note that the marker is included in the second part. Implement the following function and note the example at the bottom of the page.

```
function CA = split(M, mar)
% Split each row of matrix M into two parts at the first occurrence of the marker
%   (parameter mar); each part is stored in one cell in a row of 2-d cell array CA.
% M is a matrix of characters; assume M is not empty.
% mar is a vector of characters; assume mar is not empty.
% CA is an nr-by-2 cell array of strings, where nr is the number of rows in M.
%
% THE ONLY BUILT-IN FUNCTIONS ALLOWED ARE strcmp, size, length, cell.
% HINT: For each row, first search for the position of the marker.
```

Example solution:

```
 [nr,nc]=size(M);  n=length(mar);
 for r= 1:nr
    % Search M(r,:) for mar
    c= 1;
    while  c<=nc-n+1  &&  ~strcmp(M(r,c:c+n-1), mar)
        c= c+1;
    end
    % Assign to row r of CA
    if  c<=nc-n+1  % mar was found
        CA{r,1}= M(r,1:c-1);
        CA{r,2}= M(r,c:nc);
    else
        CA{r,1}= M(r,:);
        CA{r,2}= '';
    end
 end
```

Less efficient solution:

```
[nr,nc]=size(M);  n=length(mar);
for r= 1:nr
    idx= nc+1;
    for c= 1:nc-n+1
        if  idx==nc+1 && strcmp(M(r,c:c+n-1), mar)
            idx= c;
        end
    end
    CA{r,1}= M(r,1:idx-1);
    CA{r,2}= M(r,idx:nc);  % if idx>nc then M(r,idx:nc) gives empty string
end
```

For example, if mar is the string ‘ag’ and M is

```
['aaggagtt' ; ...
 'atttcag ' ; ...
 'ag      ' ; ...
 'aaaaaaaa' ]
```

Then CA is a 4-by-2 cell array:

Row 1: column 1 is the string ’a’, column 2 is the string ’aggagtt’

Row 2: column 1 is the string ’atttc’, column 2 is the string ’ag ’

Row 3: column 1 is the empty string, column 2 is the string ’ag   ’

Row 4: column 1 is the string ’aaaaaaa’, column 2 is the empty string