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)

Circle your lecture time: 9:05 or 11:15

Circle your discussion instructor's name:

	Tuesday	Wednesday
10:10		June Cho
11:15		Kun Dong
12:20	Susie Song	Helen Sun
1:25	Susie Song	Kun Dong
2:30	Matthew Davidow	Noam Eshed
3:35	Matthew Davidow	Noam Eshed

Instructions:

- This is a 90-minute, closed-book exam; no calculators are allowed.
- The exam is worth a total of 100 points, so it's about one point per minute!
- Read each problem completely, including any provided code, before starting it.
- Raise your hand if you have any questions.
- Use the backs of pages or ask for additional sheets of paper as necessary.
- Clarity, conciseness, and good programming style count for credit.
- If you supply multiple answers, we will grade only one.
- Use only Matlab code. No credit for code written in other programming languages.
- Write user-defined functions only if asked to do so.
- Do not write subfunctions.
- Do not use `switch`, `try`, `catch`, `break`, or `continue` statements or use built-in functions not discussed in the course.
- You may find the following Matlab predefined functions useful:
  `abs`, `sqrt`, `rem`, `min`, `max`, `floor`, `ceil`, `rand`, `zeros`, `ones`, `length`, `size`, `isempty`, `fprintf`, `disp`, `uint8`, `double`, `logical`, `char`, `strcmp`, `str2double`, `fopen`, `fclose`, `fgetl`, `feof`, `cell` `struct`

Examples:

`rem(5,2)` → 1, the remainder of 5 divided by 2

`min([9 5 6])` → 5, the minimum value of the *vector* argument

`min([9 5 6; 1 4 7])` → [1 4 6], the vector of column minima of the *matrix* argument

`floor(6.9)`, `floor(6)` → 6, rounds down to the nearest integer

`ceil(8.1)`, `ceil(9)` → 9, rounds up to the nearest integer

`[nr,nc,np]=size(M)` → dimensions of `M`: `nr` rows, `nc` columns, `np` layers

`zeros(2,4)` → a 2-by-4 matrix of zeros, type `double`

`cell(3,2)` → a 3-by-2 cell array, each cell is the empty numeric vector `[]`

`logical(1)` → `true`, convert numeric value to type `logical`. Similarly, `logical(0)` → `false`

`strcmp('cat','Cat')` → 0, the two strings are not identical

Question 1: (10 points)

(a) What are the first seven lines of output from executing the following script? If there will be fewer than seven lines, write "no output" in the space provided for each of those lines with no output.

```
n= 5;
for i = 1:n-2
    for j = i+1:n-1
        for k = j+1:n
            disp([i j k])
        end
    end
end
```

Solution:

```
Line 1:    1    2    3
Line 2:    1    2    4
Line 3:    1    2    5
Line 4:    1    3    4
Line 5:    1    3    5
Line 6:    1    4    5
Line 7:    2    3    4
```

(b) What is the result from executing the script below? Circle one answer (A, B, C, or D):

```
n= 5;
for i = 1:n
    for j = i+1:n
        for k = j+1:n
            disp([i j k])
        end
    end
end
```

A. an error message only

B. one or more lines of output followed by an error message

C. the same output as that produced by the script in Part (a)

D. output different from that produced by the script in Part (a) and no error message

Solution: $\boxed{\textbf{C}}$

Question 2: (25 points)

Implement the following functions as specified.

```
function D = allDist(P)
% Calculate the distances between the cities in struct array P.
% P: 1-d struct array of location data.  Each struct in P has these 3 fields:
%    name: (char row vector) the name of a city
%    x: (type double scalar) x-coordinate of the city
%    y: (type double scalar) y-coordinate of the city
%   P contains data for distinct locations and the length of P is at least 2.
% D: square matrix (type double) storing all pair-wise distances between the locations
%   in P.  The number of rows (and columns) of D is the length of array P.  D(i,j) is
%   the distance between locations i and j, which is equal to D(j,i).
% Only these built-in functions are allowed: length, zeros, sqrt
% For full credit, be efficient by avoiding unnecessary calculations.
```

Example solution:

```
n= length(P);
D= zeros(n,n);  % initialization not required
for i= 2:n
    for j= 1:i-1  % traverse lower triangular part of D
        D(i,j)= sqrt((P(i).x-P(j).x)^2 + (P(i).y-P(j).y)^2);
        D(j,i)= D(i,j);
    end
end
```

```
function V = furthestApart(P)
% Determine the two locations that are furthest apart given location data in P.
% P: 1-d struct array of location data as described above.
% V: Length 2 cell array storing the names of the two cities in P that are furthest apart.
%    If multiple pairs of cities are the same maximum distance apart, V is any one of
%    those pairs.
% Make effective use of function allDist from Part (a).
% Only these built-in functions are allowed: length, size, cell
% For full credit, be efficient by avoiding unnecessary iterations.
```

Example solution:

```
n= length(P);
D= allDist(P);
maxSoFar= 0;
for r= 2:n
    for c= 1:r-1
        if D(r,c) > maxSoFar
            maxSoFar= D(r,c);
            pair= [r c];
        end
    end
end
V{1}= P(pair(1)).name;
V{2}= P(pair(2)).name;
```

Question 3: (20 points)

```
function B = averageContrast(im, s, th)
% Find the blocks of an image whose "average contrast" is greater than th.
% im: 3-d array of image intensity data in type uint8.  Assume im has at least 10 rows
%   and 10 columns of pixels.
% s: the number of rows (also the number of columns) of pixels in each block.  Assume
%   s>=10, s<=nr, and s<=nc, where nr and nc are the number of rows and number of columns,
%   respectively, of im.  Partition the image into consecutive, non-overlapping, whole
%   blocks starting from the top-left pixel of the image.  For each whole block determine
%   the red contrast, green contrast, and blue contrast: each is the difference between
%   the largest and smallest intensity values of one layer of im.  The "average contrast"
%   is the mean of the red, green, and blue contrast values.  Do not calculate contrasts
%   on any incomplete blocks.
% th: a uint8 scalar
% B: a 2-d cell array where the number of rows is the number of blocks whose average
%   contrast is strictly greater than th.  In each row of B, the 1st cell stores the row
%   number of the top-left pixel of that block (type double); the 2nd cell stores the
%   column number of the top-left pixel of that block (type double); the 3rd cell stores
%   the average contrast of that block (type uint8).  If no block has an average contrast
%   greater than th, then B is an empty cell array.
% Only these built-in functions are allowed: size, floor, rem, max, min, uint8, double
```

Example solution:

```
B= {};
idx= 0;

[nr, nc, np]= size(im);
nRowsOfBlks= floor(nr/s);
nColsOfBlks= floor(nc/s);

for rStart= 1:s:nRowsOfBlks*s  % end of range is any value in
                               %   [(nRowsOfBlks-1)*s+1 .. nRowsOfBlks*s]
    for cStart= 1:s:nColsOfBlks

        currentBlk= im(rStart:rStart+s-1, cStart:cStart+s-1, :);

        aveContrast= 0;
        for layer= 1:np
            contrast= max(max(currentBlk(:,:,layer))) - min(min(currentBlk(:,:,layer)));
            aveContrast= aveContrast + contrast/3;  % in uint8
        end

        if aveContrast > th
            idx= idx + 1;
            B{idx, 1}= rStart;
            B{idx, 2}= cStart;
            B{idx, 3}= aveContrast;
        end
    end
end
```

Question 4: (25 points)

Implement the following function as specified.

```
function CA = extractNetIDs(mat)
% Extract netIDs from student data in a char matrix.
% mat: 2-d char array.  Each row of mat is comma-separated text for one student, in the
%   order name, netID, and enrolled courses.  Some rows in mat may include trailing spaces
%   such that all rows have the same length.  Each student has a name, a netID, and at
%   least one course.  mat is not empty.
% CA: 1-d cell array storing the netIDs of the students in mat; each cell stores the netID
%   (char row vector) of a student. The number of cells in CA is the number of rows in mat.
% For example, if
%   mat = ['Bynn Koh,bk327,CS1112,MATH1920          '; ...
%          'Jan Abb,jba1211,CS1110,MATH1920,CHEM2090']
% then CA is a length 2 cell array where the 1st cell stores 'bk327' and the 2nd cell
%   stores 'jba1211'.
% Only these built-in functions are allowed: length, size, cell, zeros
```

Example solution:

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

CA= cell(nr, 1);  % initialization not necessary
for r= 1:nr

    % look for first two commas
    nCommas= 0;  % number of commas so far
    c= 1;
    while  nCommas<2  % OK not to check  c<=nc  since the first 2 comma are guaranteed

        if mat(r,c)==','

            nCommas= nCommas + 1;

            idxComma(nCommas)= c;
        end
        c= c + 1;
    end

    % netID is between the first two commas
    CA{r}= mat(r, idxComma(1)+1:idxComma(2)-1 );
end
```

Question 5: (20 points)

Assume the availability of function `isIn`:

```
function tf = isIn(nameList, name)
% tf is true if name is in nameList; otherwise tf is false.
% nameList: 1-d cell array where each cell stores one name (char row vector); nameList
%   may be empty.
% name: char row vector; name is not empty.
% For example,  isIn({'Anita Borg', 'Alan Turing'}, 'Anita Borg')    returns  true
%               isIn({'Anita Borg', 'Alan Turing'}, 'Anita Turing')  returns  false
%               isIn({}, 'Anita Borg')                                returns  false
```

Do *not* implement function `isIn` but assume its availability and make effective use of it when implementing the following function:

```
function S = attendanceScore(att, studentList)
% Calculate attendance score of each student.
% att: 1-d struct array of attendance data. The length of att is the number of lectures at
%   which attendance was taken.  att is not empty. Each struct in att has two fields:
%     pres: 1-d cell array of the names of the students present and on time for a lecture.
%       Each cell stores the name (char row vector) of one student.  pres may be empty.
%     lat: 1-d cell array of the names of the students late for a lecture.  Each cell
%       stores the name (char row vector) of one student.  lat may be empty.
% studentList: 1-d cell array of the names of all students in the course; each cell stores
%   the name (char row vector) of one student.  studentList is not empty.
% S: 2-d cell array where the number of rows is the number of students in the course.  On
%   each row, the 1st cell stores a student name and the 2nd cell stores the attendance
%   score of that student.  A student earns 1 point per lecture for being on time, 0.5
%   point per lecture if s/he was late.
% Make effective use of function isIn.
```

Example solution:

```
for k= 1:length(studentList)

    S{k, 1}= studentList{k};

    S{k, 2}= 0;

    for j= 1:length(att)

        if isIn(att(j).pres, studentList{k})

            S{k,2} = S{k,2} + 1;

        elseif isIn(att(j).lat, studentList{k})

            S{k,2} = S{k,2} + 0.5;
        end
    end
end
```