Name: _____ NetID: _____
(Legibly print last name, first name, middle name)

Statement of integrity:

It is a violation of the Code of Academic Integrity to look at any exam other than your own, to look at any reference material outside of this exam packet, to communicate with anyone other than the exam proctors, or to otherwise give or receive any unauthorized help during the exam.

Academic Integrity is expected of all students of Cornell University at all times. **By submitting this exam, you declare that you will not give, use, or receive unauthorized aid in this examination.**

Circle your discussion section:

Section	Day and Time	Instructor
201	W 10:10 AM - 11:00 AM	Aravind Suresh Babu & Carlos Alvarez
202	W 11:20 AM - 12:10 PM	Carlos Alvarez
203	W 12:25 PM - 1:15 PM	Dominic Diaz
204	W 1:30 PM - 2:20 PM	Dominic Diaz
205	W 2:40 PM - 3:30 PM	Xinran Zhu
206	W 3:45 PM - 4:35 PM	Xinran Zhu

Instructions:

- Check that this packet has 8 double-sided sheets.
- 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.
- Do not modify any *given* code unless asked to do so.
- Raise your hand if you have any questions.
- Use the back of the pages if you need additional space.
- Clarity, conciseness, and good programming style count for credit.
- Indicate your final answer. If you supply multiple answers, you may receive a *zero* on that question.
- Use only MATLAB code. No credit for code written in other programming languages.
- Assume there will be no input errors.
- Write user-defined functions and subfunctions only if asked to do so.
- Do not use `switch`, `try`, `catch`, `break`, `continue`, or return statements.
- Do not use built-in functions that have not been discussed in the course.
- You may find the following MATLAB predefined functions useful: `abs`, `sqrt`, `rem`, `min`, `max`, `floor`, `ceil`, `rand`, `zeros`, `ones`, `sum`, `length`, `size`, `fprintf`, `disp`, `uint8`, `double`, `char`, `strcmp`, `str2double`, `cell`

Examples:
`zeros(1,4)` → 1 row 4 columns of zeros, type `double`
`cell(3,2)` → a 3-by-2 cell array, each cell is the empty numeric vector `[]`
`length([2 4 8])` → 3, length of a vector
`[nr,nc,np]=size(M)` → dimensions of `M`: `nr` rows, `nc` columns, `np` layers
`strcmp('cat','Cat')` → 0, the two strings are not identical
`str2double(' -2.6 ')` → −2.6, a type `double` scalar
`uint8(4.7)` → the integer (type `uint8`) value 5

Question 1 (8 points)

(1.1) 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.

```
%       1234567890
vec = 'ABCAADAEFA';
n = length(vec);
x = 0;
for k = 1:n
    if strcmp(vec(k), 'A')
        vec(n-k+1) = 'G';
        x = x + 1;
    end
end
disp(x)
```

Output: **3**

(1.2) 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 = [2 3 4 5 6 7 8; ...
     3 4 5 6 7 8 9];

[nr,nc] = size(M);
for r = nr:-1:1
    for c = 1:nc
        t = M(r,c);
        M(r,c) = M(r,nc-c+1);
        M(r,nc-c+1) = t;
    end
end
disp(M(1,:))
```

Solution: *The elements end up in their original order because the column loop goes the full range in doing swaps. The elements would be in reversed order if the column loop goes half way only in doing swaps.* Therefore the output is $\boxed{2\ 3\ 4\ 5\ 6\ 7\ 8}$

Question 2 (9 points)

Consider the function `mystery` below, which has an incomplete specification. Read the function and then complete the specification by answering questions (2.1) to (2.4).

```
function x = mystery(M)
% M is a non-empty 2-d array, where each element is an integer between 1
%   and 10, inclusive.
% x is ???

vec = zeros(1,10);
[nr, nc] = size(M);

for r = 1:nr
    for c = 1:nc
        u = M(r,c);
        vec(u) = vec(u) + 1;
    end
end
% What is vec? See Question (2.1).

b = 0;
x = 0;
for k = 1:10
    if vec(k) > b
        b = vec(k);
        x = k;
    end
end
```

(2.1) Describe variable `vec`. Hint: imagine that you are writing a comment to define `vec` at the end of the nested loops—how would you define the values in `vec`?

`vec` has length 10 and is the tally of the values in `M`. Specifically, `vec(u)` is the number of times that the integer `u` appears in `M`.

(2.2) What is the type (class) of the return parameter `x`?

`x` is of type `double`

(2.3) What is the dimension of `x`? (Is it scalar, a vector, ...?)

`x` is a scalar

(2.4) Complete the function specification by writing below a concise description for `x`.

`x` is the mode of (the value that occurs most frequently in) `M`. If there are multiple modes, `x` stores any one of them.

Question 3 (17 points)

Implement the following function as specified:

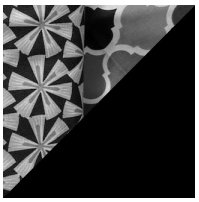
```
function Q = grayscaleTriangle(P)
% P is a non-empty 3-d uint8 array of image data. P is square: it has the
%   same number of rows and columns.
% Q is a 2-d uint8 array computed from P:
%   Q has the same number of rows and columns as P. Each pixel in the "upper
%   left triangular half" of Q has the average of the red, green, and blue
%   intensities of the corresponding pixel in P. The other pixels of Q
%   each has the uint8 value 0, corresponding to black.
% We define the "upper left triangular half" to include the minor diagonal.

[nr, nc, np]= size(P);
Q= uint8(zeros(nr,nc));
% Do not modify the code above, which assigns 0 to every element of Q.
% Add code below to assign appropriate values to the upper left triangular
% half of Q.
```

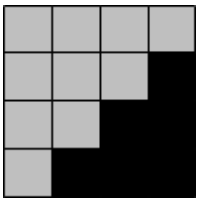
Example solution:

```
for r= 1:nr
    for c= 1:nc-r+1
        ave= P(r,c,1)/3 + P(r,c,2)/3 + P(r,c,3)/3;  % no overflow in uint8
        Q(r,c)= ave;
    end
end
```

imshow(P)

imshow(Q)





Which part is the "upper left triangular half" of a matrix? In the 4×4 matrix example on the right, the elements in the upper left triangular half are gray; the other elements are black.

Question 4 (22 points)

Implement the following function as specified:

```
function nPro = checkLetters(M)
% M is a 2-d type char matrix.  M is at least 3-by-3 in size.
% nPro is the number of "protected" characters within M. A character within
%   M is "protected" if at least half of the characters that surround it
%   are the same as itself.

% Example vectorized solution
nPro= 0;
[nr, nc]= size(M);
for r= 1:nr
    for c= 1:nc
        % At (r,c): set neighborhood H to include itself
        rmin= max(r-1,1); rmax= min(r+1,nr);
        cmin= max(c-1,1); cmax= min(c+1,nc);
        H= M(rmin:rmax, cmin:cmax);
        [nrH, ncH]= size(H);
        % Check H for same letter at M(r,c)
        nSame= sum(sum(H==M(r,c))) - 1;  % -1 to exclude itself
        if nSame >= ((nrH*ncH)-1)/2      % -1 from H size to exclude itself
            nPro= nPro + 1;
            fprintf('%c at (%d,%d)\n', M(r,c), r, c)
        end
    end
end

% Example non-vectorized solution
nPro= 0;
[nr, nc]= size(M);
for r= 1:nr
    for c= 1:nc
        % At (r,c): count neighbors and neighbors with same letter,
        % including itself for now
        nNeighbors= 0;
        nSame= 0;
        for i= r-1:r+1
            for j= c-1:c+1
                if i>0 && i<=nr && j>0 && j<=nc
                    nNeighbors= nNeighbors + 1;
                    if M(r,c)==M(i,j)
                        nSame= nSame + 1;
                    end
                end
            end
        end
        % nSame and nNeighbors include (r,c) so need to subtract 1
        if nSame-1 >= (nNeighbors-1)/2
            nPro= nPro + 1;
            fprintf('%c at (%d,%d)\n', M(r,c), r, c)
        end
    end
end
```

Question 5 (16 points)

Assume that a 1-d cell array `com` stores the names of some chemical compounds, one name in each cell. Find the first occurrence of two consecutive compounds that begin with the prefix 'Eth'. Print the index of that occurrence and the names of the two compounds. If no two consecutive cells store names with the prefix 'Eth', then print the message "not found". You can assume that all the names in `com` are at least 3 characters long. For example, if `com` is the cell array

```
{'Methane', 'Ethanol', 'Ethane', 'Butane', 'Propanol', 'Ethene'}
```

then your code should print 2, "Ethanol", and "Ethane".

For full credit, use the linear search algorithm with a `while`-loop to solve this problem. Built-in functions `find`, `strfind`, and `findstr` are forbidden.

```
% Assume 1-d cell array com, as described above, is given.  com is not empty.
% Write your code below.
```

Example solution:

```
pre= 'Eth';
n= length(com);
k= 1;
while k <= n-1 && (~strcmp(com{k}(1:3),pre) || ~strcmp(com{k+1}(1:3),pre))
    k= k + 1;
end
if  k <= n-1  % found
    disp(k), disp(com{k}), disp(com{k+1})
else
    disp('not found')
end
```

Example solution:

```
pre= 'Eth';
n= length(com);
k= 1;
found= false;
while k <= n-1 && ~found
    if strcmp(com{k}(1:3),pre) && strcmp(com{k+1}(1:3),pre)
        found= true;
        disp(k), disp(com{k}), disp(com{k+1})
    end
    k= k + 1;
end
if ~found
    disp('not found')
end
```

Question 6 (28 points)

For each exercise in MATLAB Grader, we get one data file containing students' per-problem scores. Here, we assume that each exercise always has exactly two problems. Each line in the file contains the data for one student: email address, problem 1 score, problem 2 score. Below is an example showing the format of the data file.

```
xyz756@cornell.edu,0.13,0.5
abc123@cornell.edu,0.3,--
ef9@cornell.edu,0.5,0
ghi18@cornell.edu,--,0.1
omg2151@cornell.edu,--,--
```

The characters '--' indicate that a student did not submit an answer to a problem. A student's submission is incomplete if the data for that student is '--' for one or both problems. Note that a 0 for a problem in the data file is not the same as incomplete. Therefore the example above indicates 2 complete and 3 incomplete submissions.

Complete the function `checkSubmissions` on the next page to parse the data file and compute each student's exercise score as specified. Assume the file has at least one line of data. If a student did not submit a problem (indicated by '--'), assign a numeric score of 0 for that problem.

You can add code above, below, and between the given statements, but do not cross out any given code. Built-in functions `find`, `strfind`, and `findstr` are forbidden.

```
fid= fopen(inFilename, 'r');

nIncomp= 0;  % #incomplete submissions
s= 0;        % #student data lines read so far
while ~feof(fid)
    s= s+1;
    line= fgetl(fid);
    info= textscan(line, '%s %s %s', 'delimiter',',');
    info= cellstr2str(info);  % remove nesting of cells
    % info is a 1-d cell array of length 3; each cell stores a char vector,
    %   e.g., {'abc123@cornell.edu', '0.3', '--'}

    % get NetID from from info{1}
    email= info{1};
    k= 1;
    while email(k)~='@'  % No need to check k since data assumed correct
        k= k+1;
    end
    sData{s,1}= email(1:k-1);

    % get scores and update stats
    isIncomplete= 0;  % assume complete for each student to start
    for c= 2:3
        if ~strcmp(info{c},'--')
            sData{s,c}= str2double(info{c});
        else
            sData{s,c}= 0;
            isIncomplete= 1;
        end
    end
    sData{s,4}= sData{s,2} + sData{s,3};
    nIncomp= nIncomp + isIncomplete;
end

fprintf('Processed data for Exercse %d.', exNum)
fprintf(' %d submissions in total,', s)
fprintf(' %d submissions incomplete.\n', nIncomp)
fclose(fid);
```