

CS 1112 Prelim 2 Review

Prelim 2 Topics

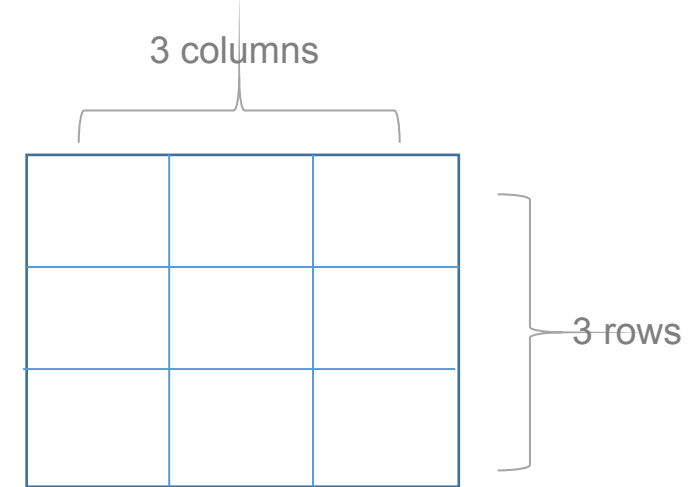
- 2-dimensional arrays (matrices)
 - 2-d array traverse with nested loops
 - Partial matrix traverse, e.g., triangular
- 3-dimensional arrays (e.g., color image data)
- type uint8 data (doing arithmetic with uint8 variables)
- vectorized code
- character and char arrays (1-d, 2-d) (We do not use type String. No ASCII arithmetic.)
- cell arrays
- Linear search

2-D and 3-D arrays

2-D array (also known as a matrix)

- A collection of data (e.g. numbers, or characters, but not both) stored in rows and columns
- Items are referenced using 2 numbers (row # and column #) $A(i,j)$

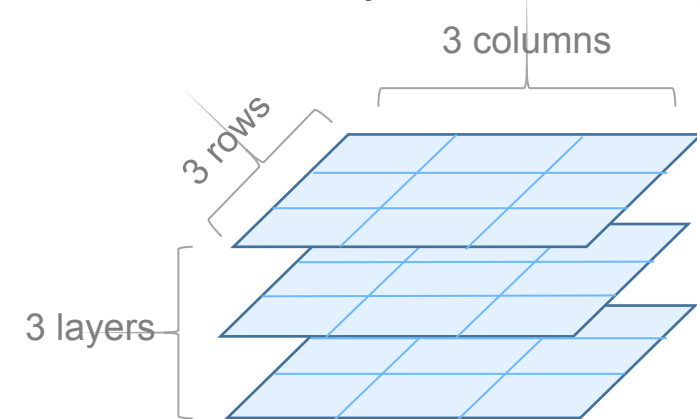
Example: 3x3 matrix



3-D array

- A series of 2-D arrays layered on each other
- Items are referenced by 3 numbers (row #, column #, layer #) $A(i,j,k)$

Example: 3x3x3 3-D array



2-D and 3-D arrays

Initialize a 2-D array, A

% Create 3x3 matrix of zeros

```
A = zeros(3,3);
```

Find size of 2-D array A

```
[nr,nc] = size(A); % nr=3,nc=3
```

Pattern for *traversing* a 2-D array

```
for r = 1:nr
```

```
    for c = 1:nc
```

```
        % Do something to A(r,c)
```

```
    end
```

```
end
```

Initialize a 3-D array, B

% Create 3x3x3 matrix of zeros

```
B = zeros(3,3,3);
```

Find size of 3-D array B

```
[nr,nc,np] = size(B); % nr=3,nc=3,np=3
```

Note: this would also work for a 2-D array; np would just be 1.

Pattern for *traversing* a 3-D array

```
for r = 1:nr
```

```
    for c = 1:nc
```

```
        for p = 1:np
```

```
            % Do something to A(r,c,p)
```

```
        end
```

```
    end
```

```
end
```

Remember the image of a 3-D array as a *stack* of 2-D arrays: this pattern works by examining *down* the layers, then *across* the columns, then *down* the rows

Matrix transposition

Given a matrix M, where

$$M = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

We'd like to create a matrix T, which stores the *transpose* of M.

$$T = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

How do we write code to do this?

Hints: the *rows* of M have become the *columns* of T. M(1, 2) is the same as T(2, 1).

Solution:

- We know that we need a *nested for-loop* to go through all elements of M.
- Relation between M and T: reverse positions in M to get positions in T

```
[nr,nc] = size(M);  
for r = 1:nr  
    for c = 1:nc  
        T(c,r) = M(r,c);  
    end  
end
```

Mirror image of a 2-D array

Given a matrix M , where

$M = \begin{matrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{matrix}$

We'd like to create a matrix T , which is the *mirror image* of M .

$T = \begin{matrix} 4 & 3 & 2 & 1 \\ 8 & 6 & 7 & 5 \end{matrix}$

How do we write code to do this?

Hint: Reverse the order of the columns of M to get T .

How do we re-order the columns?

$M = \begin{matrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{matrix}$

Switch columns 2 and $nc-1$

Switch columns 1 and nc

Relationship between a pair of columns that must be reversed:

Column c should be replaced by column $(nc - c) + 1$.

Mirror image of a 2-D array

Given a matrix M, where

$$M = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{bmatrix}$$

We'd like to create a matrix T, which is the *mirror image* of M.

$$T = \begin{bmatrix} 4 & 3 & 2 & 1 \\ 8 & 6 & 7 & 5 \end{bmatrix}$$

How do we write code to do this?

Hint: Reverse the order of the columns of M to get T.

Non-vectorized solution:

Column c should be exchanged with column $(nc - c) + 1$.

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

Mirror image of a 2-D array

Given a matrix M, where

$$M = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{bmatrix}$$

We'd like to create a matrix T, which is the *mirror image* of M.

$$T = \begin{bmatrix} 4 & 3 & 2 & 1 \\ 8 & 6 & 7 & 5 \end{bmatrix}$$

How do we write code to do this?

Hint: Reverse the order of the columns of M to get T.

Vectorized solution:

Column c should be exchanged with column $(nc - c) + 1$.

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

The vectorized solution exchanges entire columns instead of individual elements.

Mirror image of a 3-D array

Non-vectorized solution:

Column c should be exchanged with column $(nc - c) + 1$.

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

Vectorized solution:

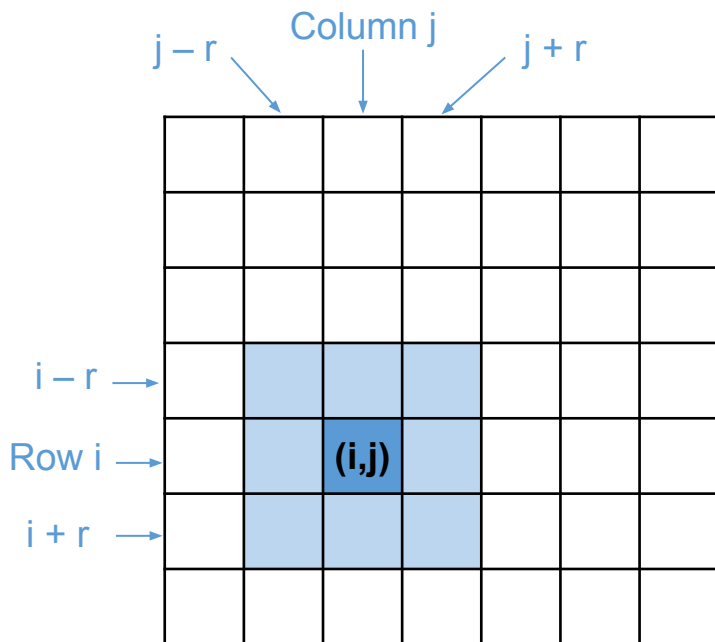
Column c should be exchanged with column $(nc - c) + 1$.

```
[nr, nc, np] = size(M);  
for c = 1:nc  
    T(:,c,:) = M(:,nc-c+1,:);  
end
```

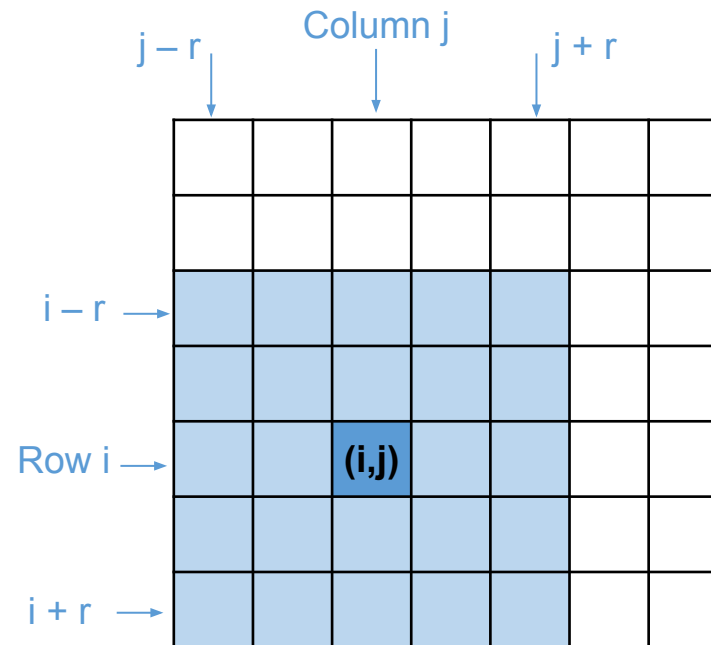
2-D and 3-D arrays: Sub-arrays

The **neighborhood** of a particular cell in an array is the set of cells that *surround* that one cell within a particular radius.

Neighborhood radius: $r = 1$



Neighborhood radius: $r = 2$



How do we *extract the sub-array* that corresponds to the highlighted neighborhood?

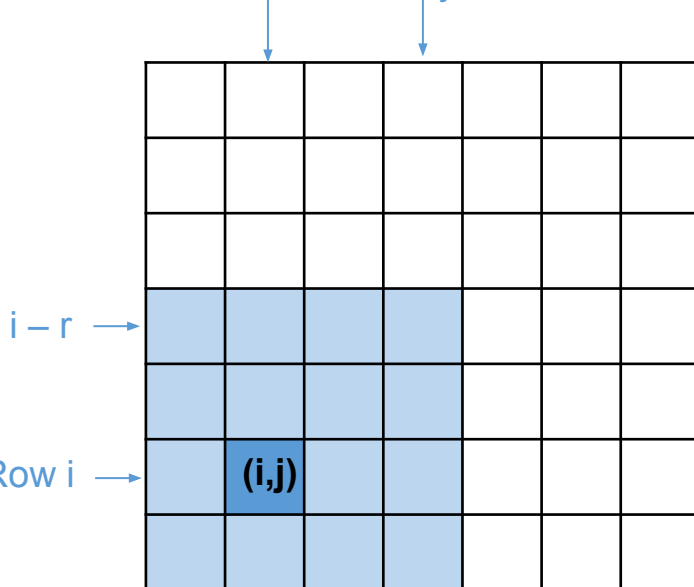
We need to select all rows between row $i-r$ and row $i+r$, and all columns between columns $j-r$ and $j+r$.

2-D and 3-D arrays: Sub-arrays

From previous slide: We need to select all rows between row $i-r$ and row $i+r$, and all columns between columns $j-r$ and $j+r$. What if the neighborhood goes out of bounds?

Neighborhood radius: $r = 2$

Column j $j+r$



How do we **generalize** this approach so that it works when $i-r$ or $i+r$ or $j-r$ or $j+r$ are out of bounds?

Solution:

```
[m,n] = size(M);  
iMin = max(1,i-r)  
iMax = min(m,i+r)  
jMin = max(1,j-r)  
jMax = min(n,j+r)
```

```
% Now extract submatrix: the neighborhood  
C = M(iMin:iMax, jMin:jMax)
```

2-D and 3-D arrays: Resizing an array

Interpolating on a matrix means:

- Inserting new rows/columns between existing rows/columns
- The new rows/columns are calculated from a pair of originally adjacent rows/columns

Example of expanding a matrix (original cells in gray):

1	2	3
4	5	6

1	1.5	2	2.5	3
2.5	3	3.5	4	4.5
4	4.5	5	5.5	6

2-D and 3-D arrays: Resizing an array

If the interpolation involves creating additional rows *and* columns, it is easier to work with one dimension at a time, i.e. columns first and then rows.

Example of expanding a matrix (original cells in gray):

1	2	3
4	5	6

Size: $nr \times nc$



1	1.5	2	2.5	3
4	4.5	5	5.5	6

Size: $nr \times 2*nc-1$



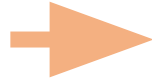
1	1.5	2	2.5	3
2.5	3	3.5	4	4.5
4	4.5	5	5.5	6

Size: $2*nr-1 \times 2*nc-1$

2-D and 3-D arrays: Resizing an array

1	2	3
4	5	6

Size: $nr \times nc$



1	1.5	2	2.5	3
4	4.5	5	5.5	6

Size: $nr \times 2*nc-1$



1	1.5	2	2.5	3
2.5	3	3.5	4	4.5
4	4.5	5	5.5	6

Size: $2*nr-1 \times 2*nc-1$

Step 1: interpolate on columns

```
for r= 1:nr
```

```
    for c= 1:nc-1
```

```
        % copy original data
```

```
        wideM(r,c*2-1)= M(r,c);
```

```
        % calculate average of adjacent columns
```

```
        wideM(r,c*2)= M(r,c)/2 + M(r,c+1)/2;
```

```
    end
```

```
    wideM(r,nc*2-1)= M(r,nc);
```

```
end
```

Step 2: interpolate on rows

```
for c= 1:nc*2-1
```

```
    for r= 1:nr-1
```

```
        % copy data from intermediate matrix
```

```
        newM(r*2-1,c)= wideM(r,c);
```

```
        % calculate average of adjacent rows
```

```
        newM(r*2,c)= wideM(r,c)/2 + wideM(r+1,c)/2;
```

```
    end
```

```
    newM(nr*2-1,c)= wideM(nr,c);
```

```
end
```

2-D and 3-D arrays: Resizing an array

How might we **reduce** a 2-D array by averaging each group of 4 cells?

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16



3.5	5.5
11.5	13.5

Solution:

- Size of new matrix: $\frac{1}{2}$ the size of the original one
- We can *pick one cell* in each group (for example, pick the cells containing the values 6, 8, 14, 16), and calculate the average of it and its *neighbors*.

```
[nr,nc] = size(M);  
N = zeros(nr/2, nc/2);  
for r = 2:2:nr  
    for c = 2:2:nc  
        avg = (M(r-1,c)+M(r-1,c-1)+M(r,c-1)+M(r,c))/4;  
        N(r/2,c/2) = avg;  
    end  
end
```

Fall 2020 Prelim: Question 2

Question 2 (15 points)

Implement the following function as specified.

```
function v = countPositives(D)
% Count the number of positive values in each column in the lower left
%   triangular part of D, including the main diagonal.
% D: an n-by-n matrix where n>1
% v: a row vector of length n. The kth value in v is the number of
%   positive values in the kth column of D at and below the main diagonal.
% Example: If D is [ 9  2  1  0; ...
%                  -1 -3  0  0; ...
%                  4 -5  7  6; ...
%                  0  0  3  8]
%
%           then v is [ 2  0  2  1 ]
```


Fall 2020 Prelim: Question 2

```
function v = countPositives ( D )
```

```
end
```

Fall 2020 Prelim: Question 2

```
function v = countPositives ( D )  
[n,~] = size(D);  
v = zeros(1,n);
```

```
end
```

Fall 2020 Prelim: Question 2

```
function v = countPositives ( D )
```

```
[n,~] = size(D);
```

```
v = zeros(1,n);
```

```
for r = 1:n
```

```
    for c = __ : __
```

```
        end
```

```
    end
```

```
end
```

Fall 2020 Prelim: Question 2

```
function v = countPositives ( D )
```

```
[n,~] = size(D);
```

```
v = zeros(1,n);
```

```
for r = 1:n
```

```
    for c = 1 : r
```

```
        if D(r,c) > 0
```

```
            v(c) = v(c) + 1
```

```
        end
```

```
    end
```

```
end
```

```
end
```

Fall 2020 Prelim: Question 2

Designing an algorithm

#	Thing we need to do	Programming concept needed to do this thing
1	Create a 2D array of the correct size	
2		
3		

Fall 2020 Prelim: Question 2

Designing an algorithm

#	Thing we need to do	Programming concept needed to do this thing
1	Create a 2D array of the correct size	Use the size() and zeros() functions to get the size and make the array
2		
3		

Fall 2020 Prelim: Question 2

Designing an algorithm

#	Thing we need to do	Programming concept needed to do this thing
1	Create a 2D array of the correct size	Use the size() and zeros() functions to get the size and make the array
2	Iterate through the part of the array we want	
3		

Fall 2020 Prelim: Question 2

Designing an algorithm

#	Thing we need to do	Programming concept needed to do this thing
1	Create a 2D array of the correct size	Use the size() and zeros() functions to get the size and make the array
2	Iterate through the part of the array we want	Create a nested for-loop with the right loop parameter
3	Keep track of the number of positive values seen so far	

Fall 2020 Prelim: Question 2

Designing an algorithm

#	Thing we need to do	Programming concept needed to do this thing
1	Create a 2D array of the correct size	Using the size() and zeros() functions to get the size and make the array
2	Iterate through the part of the array we want	Creating a nested for-loop with the right loop parameter
3	Keep track of the number of positive values seen so far	Use if-statement separate cases, and then incrementing array values

Matlab data types

A **type** is a way of representing data. You should be aware of these types:

Matlab data types

A **type** is a way of representing data. You should be aware of these types:

- **double:** the default type for numbers in Matlab
Array of doubles: `x = [1, 2, 3];`

Matlab data types

A **type** is a way of representing data. You should be aware of these types:

- **double:** the default type for numbers in Matlab
Array of doubles: `x = [1, 2, 3];`
- **uint8:** integers ranging from 0 to 255
Array of uint8 numbers: `y = uint8(x);`

Matlab data types

A **type** is a way of representing data. You should be aware of these types:

- **double:** the default type for numbers in Matlab
Array of doubles: `x = [1, 2, 3];`
- **uint8:** integers ranging from 0 to 255
Array of uint8 numbers: `y = uint8(x);`
- **char:** standard characters, including letters, digits, symbols. Multiple chars together form an array of characters.
Array of characters: `s = 'CS1112'; s = ['c', 's', '1', '1', '1', '2'];`

Matlab data types

A **type** is a way of representing data. You should be aware of these types:

- **double:** the default type for numbers in Matlab
Array of doubles: `x = [1, 2, 3];`
- **uint8:** integers ranging from 0 to 255
Array of uint8 numbers: `y = uint8(x);`
- **char:** standard characters, including letters, digits, symbols. Multiple chars together form an array of characters.
Array of characters: `s = 'CS1112'; s = ['c', 's', '1', '1', '1', '2'];`
- **logical:** also known as a boolean. Can be true/false or 0/1.
Creating a logical: `z = rand > 0.5`

Matlab data types

A **type** is a way of representing data. You should be aware of these types:

- **double:** the default type for numbers in Matlab
Array of doubles: `x = [1, 2, 3];`
- **uint8:** integers ranging from 0 to 255
Array of uint8 numbers: `y = uint8(x);`
- **char:** standard characters, including letters, digits, symbols. Multiple chars together form an array of characters.
Array of characters: `s = 'CS1112'; s = ['c', 's', '1', '1', '1', '2'];`
- **logical:** also known as a boolean. Can be true/false or 1/0.
Creating a logical: `z = rand > 0.5`

An array can only hold values of one type. A *cell array* is a special kind of array that can hold data of different types. Yay!

Working with images: uint8 type

What is uint8?

An integer that can hold values between 0 and 255 ($2^8 - 1 = 255$). Images can be represented with numbers in this range.

Working with images: uint8 type

What is uint8?

An integer that can hold values between 0 and 255 ($2^8 - 1 = 255$). Images can be represented with numbers in this range.

How to convert numeric data into uint8:

Given an array x of (regular) numbers,

$y = \text{uint8}(x);$

Working with images: uint8 type

What is uint8?

An integer that can hold values between 0 and 255 ($2^8 - 1 = 255$). Images can be represented with numbers in this range.

How to convert numeric data into uint8:

Given an array `x` of (regular) numbers,

```
y = uint8(x);
```

Note about overflow:

If you need to perform an arithmetic operation on uint8 numbers, e.g. averaging, be careful that the numbers won't overflow, i.e. exceed 255.

```
goodAverage = x(1)/3 + x(2)/3 + x(3)/3;    % Do this
```

```
badAverage = (x(1) + x(2) + x(3))/3;    % Don't do this
```

Fall 2019 Prelim: Question 4

Question 4 (15 points)

In class you have worked with images in the *RGB* colorspace, where channel 1 is *R* (red), channel 2 is *G* (green), and channel 3 is *B* (blue). But images can be represented in other colorspace. Consider the *YCoCg* colorspace, where the channels *Y*, *C_o*, and *C_g* are mathematically related to *R*, *G*, and *B* via the following equations:

$$Y = \frac{1}{4}(2G + R + B)$$
$$C_o = \frac{1}{2}(R - B) + 128$$
$$C_g = \frac{1}{4}(2G - R - B) + 128$$

(this assumes that *R*, *G*, and *B* are in the range 0-255 and will yield *Y*, *C_o*, and *C_g* in that same range).

Implement the following function (hint: remember the rules of uint8 arithmetic):

```
function ycocg = rgb2ycocg(rgb)
% Transform image from RGB colorspace to YCoCg colorspace.
% rgb is a 3D uint8 array such that rgb(i,j,k) is the value of channel
%   k (R, G, or B) for the pixel at row i and column j.
% ycocg is a 3D uint8 array such that ycocg(i,j,k) is the value of
%   channel k (Y, Co, or Cg) for the pixel at row i and column j after
%   being transformed from RGB to YCoCg.
```

Fall 2019 Prelim: Question 4

```
function ycocg = rgb2ycocg(rgb)
% Transform image from RGB colorspace to YCoCg colorspace.
% rgb is a 3D uint8 array such that rgb(i,j,k) is the value of channel
% k (R, G, or B) for the pixel at row i and column j.
% ycocg is a 3D uint8 array such that ycocg(i,j,k) is the value of
% channel k (Y, Co, or Cg) for the pixel at row i and column j after
% being transformed from RGB to YCoCg.
```

$$Y = \frac{1}{4}(2G + R + B)$$
$$C_o = \frac{1}{2}(R - B) + 128$$
$$C_g = \frac{1}{4}(2G - R - B) + 128$$

```
end
```

Fall 2019 Prelim: Question 4

```
function ycocg = rgb2ycocg(rgb)
% Transform image from RGB colorspace to YCoCg colorspace.
% rgb is a 3D uint8 array such that rgb(i,j,k) is the value of channel
% k (R, G, or B) for the pixel at row i and column j.
% ycocg is a 3D uint8 array such that ycocg(i,j,k) is the value of
% channel k (Y, Co, or Cg) for the pixel at row i and column j after
% being transformed from RGB to YCoCg.
[nr, nc, ~] = size(rgb);
ycocg = uint8(zeros(nr, nc, 3));

end
```

$$Y = \frac{1}{4}(2G + R + B)$$
$$C_o = \frac{1}{2}(R - B) + 128$$
$$C_g = \frac{1}{4}(2G - R - B) + 128$$

Fall 2019 Prelim: Question 4

```
function ycocg = rgb2ycocg(rgb)
% Transform image from RGB colorspace to YCoCg colorspace.
% rgb is a 3D uint8 array such that rgb(i,j,k) is the value of channel
% k (R, G, or B) for the pixel at row i and column j.
% ycocg is a 3D uint8 array such that ycocg(i,j,k) is the value of
% channel k (Y, Co, or Cg) for the pixel at row i and column j after
% being transformed from RGB to YCoCg.
[nr, nc, ~] = size(rgb);
ycocg = uint8(zeros(nr, nc, 3));

for r = 1:nr
    for c = 1:nc

        Y = (2*rgb(r,c,1) + rgb(r,c,2) + rgb(r,c,3))/4;
        Co = (rgb(r,c,1) - rgb(r,c,3))/2 + 128;
        Cg = (2*rgb(r,c,2) - rgb(r,c,1) - rgb(r,c,3))/4 + 128;

        ycocg(r,c,1) = Y;
        ycocg(r,c,2) = Co;
        ycocg(r,c,3) = Cg;
    end
end
end
```

$$Y = \frac{1}{4}(2G + R + B)$$
$$C_o = \frac{1}{2}(R - B) + 128$$
$$C_g = \frac{1}{4}(2G - R - B) + 128$$

Fall 2019 Prelim: Question 4

```
function ycocg = rgb2ycocg(rgb)
```

```
[nr, nc, ~] = size(rgb);  
ycocg = uint8(zeros(nr, nc, 3));
```

```
for r = 1:nr  
    for c = 1:nc
```

```
        end  
    end  
end
```

$$Y = \frac{1}{4}(2G + R + B)$$

$$C_o = \frac{1}{2}(R - B) + 128$$

$$C_g = \frac{1}{4}(2G - R - B) + 128$$

Fall 2019 Prelim: Question 4

```
function ycocg = rgb2ycocg(rgb)
[nr, nc, ~] = size(rgb);
ycocg = uint8(zeros(nr, nc, 3));

for r = 1:nr
    for c = 1:nc

        end
    end
end
```

$$Y = \frac{1}{4}(2G + R + B)$$

$$C_o = \frac{1}{2}(R - B) + 128$$

$$C_g = \frac{1}{4}(2G - R - B) + 128$$

Fall 2019 Prelim: Question 4

```
function ycocg = rgb2ycocg(rgb)
[nr, nc, ~] = size(rgb);
ycocg = uint8(zeros(nr, nc, 3));
```

```
for r = 1:nr
    for c = 1:nc
```

```
        R = rgb(r, c, 1);
        G = rgb(r, c, 2);
        B = rgb(r, c, 3);
```

```
        ycocg(r, c, 1) =
        ycocg(r, c, 2) =
        ycocg(r, c, 3) =
```

```
% Y value of current pixel
% Co value of current pixel
% Cg value of current pixel
```

```
    end
```

```
end
```

```
end
```

$$Y = \frac{1}{4}(2G + R + B)$$

$$C_o = \frac{1}{2}(R - B) + 128$$

$$C_g = \frac{1}{4}(2G - R - B) + 128$$

Fall 2019 Prelim: Question 4

```
function ycocg = rgb2ycocg(rgb)
[nr, nc, ~] = size(rgb);
ycocg = uint8(zeros(nr, nc, 3));

for r = 1:nr
    for c = 1:nc

        R = rgb(r, c, 1);
        G = rgb(r, c, 2);
        B = rgb(r, c, 3);

        ycocg(r, c, 1) = (2*G + R + B)/4;
        ycocg(r, c, 2) = (R - B)/2 + 128;
        ycocg(r, c, 3) = (2*G - R - B)/4 + 128;

    end
end
end
```

$$Y = \frac{1}{4}(2G + R + B)$$
$$C_o = \frac{1}{2}(R - B) + 128$$
$$C_g = \frac{1}{4}(2G - R - B) + 128$$

% Y value of current pixel
% Co value of current pixel
% Cg value of current pixel

Fall 2019 Prelim: Question 4

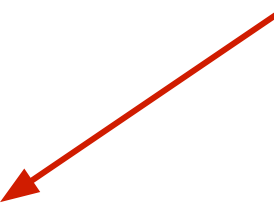
```
function ycocg = rgb2ycocg(rgb)
[nr, nc, ~] = size(rgb);
ycocg = uint8(zeros(nr, nc, 3));
```

```
for r = 1:nr
    for c = 1:nc
```

```
        R = rgb(r, c, 1);
        G = rgb(r, c, 2);
        B = rgb(r, c, 3);
```

```
        ycocg(r, c, 1) = (2*G + R + B)/4;
        ycocg(r, c, 2) = (R - B)/2 + 128;
        ycocg(r, c, 3) = (2*G - R - B)/4 + 128;
```

These formulas would lead
to overflow and underflow



% Y value of current pixel
% Co value of current pixel
% Cg value of current pixel

```
    end
```

```
end
end
```

$$Y = \frac{1}{4}(2G + R + B)$$
$$C_o = \frac{1}{2}(R - B) + 128$$
$$C_g = \frac{1}{4}(2G - R - B) + 128$$

Fall 2019 Prelim: Question 4

```
function ycocg = rgb2ycocg(rgb)
[nr, nc, ~] = size(rgb);
ycocg = uint8(zeros(nr, nc, 3));
```

```
for r = 1:nr
    for c = 1:nc
```

```
        R = double(rgb(r, c, 1));
        G = double(rgb(r, c, 2));
        B = double(rgb(r, c, 3));
```

```
        ycocg(r, c, 1) =
        ycocg(r, c, 2) =
        ycocg(r, c, 3) =
```

```
% Y value of current pixel
% Co value of current pixel
% Cg value of current pixel
```

```
    end
```

```
end
```

```
end
```

$$Y = \frac{1}{4}(2G + R + B)$$

$$C_o = \frac{1}{2}(R - B) + 128$$

$$C_g = \frac{1}{4}(2G - R - B) + 128$$

Fall 2019 Prelim: Question 4

```
function ycocg = rgb2ycocg(rgb)
[nr, nc, ~] = size(rgb);
ycocg = uint8(zeros(nr, nc, 3));
```

```
for r = 1:nr
    for c = 1:nc
```

```
        R = double(rgb(r, c, 1));
        G = double(rgb(r, c, 2));
        B = double(rgb(r, c, 3));
```

```
        ycocg(r, c, 1) = uint8((2*G + R + B)/4); % Y value of current pixel
        ycocg(r, c, 2) = uint8((R - B)/2 + 128); % Co value of current pixel
        ycocg(r, c, 3) = uint8((2*G - R - B)/4 + 128); % Cg value of current pixel
```

```
    end
```

```
end
```

```
end
```

$$Y = \frac{1}{4}(2G + R + B)$$

$$C_o = \frac{1}{2}(R - B) + 128$$

$$C_g = \frac{1}{4}(2G - R - B) + 128$$

Fall 2019 Prelim: Question 4

```
function ycocg = rgb2ycocg(rgb)
[nr, nc, ~] = size(rgb);
ycocg = uint8(zeros(nr, nc, 3));
```

```
for r = 1:nr
    for c = 1:nc
```

```
        R = double(rgb(r, c, 1));
        G = double(rgb(r, c, 2));
        B = double(rgb(r, c, 3));
```

```
        ycocg(r, c, 1) = uint8((2*G + R + B)/4);
        ycocg(r, c, 2) = uint8((R - B)/2 + 128);
        ycocg(r, c, 3) = uint8((2*G - R - B)/4 + 128);
```

```
    end
```

```
end
```

```
end
```

Make sure all numbers used
in formulas are of type
double

Solve formulas using type
double, and then convert to
uint8 at the end

% Y value of current pixel
% Co value of current pixel
% Cg value of current pixel

$$Y = \frac{1}{4}(2G + R + B)$$
$$C_o = \frac{1}{2}(R - B) + 128$$
$$C_g = \frac{1}{4}(2G - R - B) + 128$$

Fall 2019 Prelim: Question 4

```
function ycocg = rgb2ycocg(rgb)
[nr, nc, ~] = size(rgb)
ycocg = zeros(nr, nc, 3)
```

```
for r = 1:nr
    for c = 1:nc
```

```
        R = rgb(r, c, 1);
        G = rgb(r, c, 2);
        B = rgb(r, c, 3);
```

```
        ycocg(r, c, 1) = (G/2) + (R/4) + (B/2);    % Y value of current pixel
        ycocg(r, c, 2) = 128 - (B/2) - (R/2);      % Co value of current pixel
        ycocg(r, c, 3) = 128 + (G/2) - (R/4) - (B/4); % Cg value of current pixel
```

```
    end
```

```
end
```

```
end
```

$$Y = \frac{1}{4}(2G + R + B)$$

$$C_o = \frac{1}{2}(R - B) + 128$$

$$C_g = \frac{1}{4}(2G - R - B) + 128$$

Fall 2019 Prelim: Question 4

```
function ycocg = rgb2ycocg(rgb)
[nr, nc, ~] = size(rgb)
ycocg = zeros(nr, nc, 3)
```

```
R = rgb(:, :, 1);    % R panel
G = rgb(:, :, 2);    % G panel
B = rgb(:, :, 3);    % B panel
```

```
ycocg(:, :, 1) = (G/2) + (R/4) + (B/2);    % Y panel
ycocg(:, :, 2) = 128 - (B/2) - (R/2);      % Co panel
ycocg(:, :, 3) = 128 + (G/2) - (R/4) - (B/4); % Cg panel
```

```
end
```

$$Y = \frac{1}{4}(2G + R + B)$$

$$C_o = \frac{1}{2}(R - B) + 128$$

$$C_g = \frac{1}{4}(2G - R - B) + 128$$

Fall 2019 Prelim: Question 4

```
function ycocg = rgb2ycocg(rgb)
[nr, nc, ~] = size(rgb)
ycocg = zeros(nr, nc, 3)
```

```
R = double(rgb(:, :, 1));    % R panel
G = double(rgb(:, :, 2));    % G panel
B = double(rgb(:, :, 3));    % B panel
```

```
ycocg(:, :, 1) = uint8((2*G + R + B)/4);    % Y value of current pixel
ycocg(:, :, 2) = uint8((R - B)/2 + 128);    % Co value of current pixel
ycocg(:, :, 3) = uint8((2*G - R - B)/4 + 128);    % Cg panel
```

```
end
```

$$Y = \frac{1}{4}(2G + R + B)$$

$$C_o = \frac{1}{2}(R - B) + 128$$

$$C_g = \frac{1}{4}(2G - R - B) + 128$$

Fall 2019 Prelim: Question 4

Designing an algorithm

#	Thing we need to do	Programming concept needed to do this thing
1	Initialize output array to be the right size	
2		
3		

Fall 2019 Prelim: Question 4

Designing an algorithm

#	Thing we need to do	Programming concept needed to do this thing
1	Initialize output array to be the right size	Creating and getting the size of an array using <code>zeros()</code> and <code>size()</code>
2		
3		

Fall 2019 Prelim: Question 4

Designing an algorithm

#	Thing we need to do	Programming concept needed to do this thing
1	Initialize output array to be the right size	Creating and getting the size of an array using <code>zeros()</code> and <code>size()</code>
2	Go through each pixel in the arrays	
3		

Fall 2019 Prelim: Question 4

Designing an algorithm

#	Thing we need to do	Programming concept needed to do this thing
1	Initialize output array to be the right size	Creating and getting the size of an array using <code>zeros()</code> and <code>size()</code>
2	Go through each pixel in the arrays	Traversing arrays using nested for-loops
3		

Fall 2019 Prelim: Question 4

Designing an algorithm

#	Thing we need to do	Programming concept needed to do this thing
1	Initialize output array to be the right size	Creating and getting the size of an array using <code>zeros()</code> and <code>size()</code>
2	Go through each pixel in the arrays	Traversing arrays using nested for-loops
3	Calculate new uint8 values for output	

Fall 2019 Prelim: Question 4

Designing an algorithm

#	Thing we need to do	Programming concept needed to do this thing
1	Initialize output array to be the right size	Creating and getting the size of an array using <code>zeros()</code> and <code>size()</code>
2	Go through each pixel in the arrays	Traversing arrays using nested for-loops
3	Calculate new uint8 values for output	uint8 arithmetic: either convert to double and back to uint8 or rearrange formula to avoid overflow/underflow

Character and char arrays

- A char array is a 1-D array (vector) of characters, 1 character per position

'm'	'a'	't'	'l'	'a'	'b'
-----	-----	-----	-----	-----	-----

1-D array of characters (1 row, 6 columns)



```
A = 'matlab';  
A = ['m','a','t','l','a','b'];  
disp(A(3))  % prints 't'
```


Character and char arrays

- A char array is a 1-D array (vector) of characters, 1 character per cell
- A 2-D char array with n rows could store n strings, with one string per row, as long as each row has the same number of characters (columns).

'm'	'a'	't'	'l'	'a'	'b'
-----	-----	-----	-----	-----	-----

1-D array of characters (1 row, 6 columns)



```
A = 'matlab';  
A = ['m','a','t','l','a','b'];  
disp(A(3))  % prints 't'
```

Character and char arrays

- A char array is a 1-D array (vector) of characters, 1 character per cell
- A 2-D char array with n rows could store n **strings**, with one string per row, as long as each row has the same number of characters (columns).

'm'	'a'	't'	'l'	'a'	'b'
-----	-----	-----	-----	-----	-----

1-D array of characters (1 row, 6 columns)



```
A = 'matlab';  
A = ['m','a','t','l','a','b'];  
disp(A(3)) % prints 't'
```

'm'	'a'	't'	'l'	'a'	'b'
'i'	's'	' '	' '	' '	' '
'f'	'u'	'n'	' '	' '	' '

2-D array of characters (3 rows, 6 columns)



```
B = ['matlab'; 'is  '; 'fun  '];  
disp(B(1,:)) % prints 'matlab'
```

A **string** here means a **sequence of characters**, not the type string!

Note that empty spaces have to be appended onto the shorter words so that the strings on each row have the same length.

Character and char arrays: Useful functions

- **strcmp(str1, str2)**

Compares if str1 and str2 are identical.

Returns true if str1 has all the same characters as str2, and 0 if not.

```
strcmp('matlab', 'mAtlab') % 0
```

Character and char arrays: DO NOT USE ==!

Note that comparing char arrays directly with '==', e.g. `str1 == str2`, will return **an array of true false values**, where each element of `str1` is compared to the corresponding element of `str2`:

`'cat' == 'dog'` gives
`[false false false]`

or will produce a runtime error if `str1, str2` are not of same length.

Character and char arrays: Useful functions

- **strcmp(str1, str2)**

Returns 1 if str1 has all the same characters as str2, and 0 if not.

Case-sensitive.

```
strcmp('matlab', 'mAtlab') % 0
```

- **str2double(str1)**

If str1 is a char array containing a number, the function returns that number as a numerical value. Returns NaN if str1 is something other than a number.

```
str2double('20') + 2 % 22  
str2double('hi') % NaN
```

Character and char arrays: Example 1

Given a char array s, where

s = 'hello'

We'd like to reverse it to obtain

r = 'olleh'

How to we write code to do this?

Character and char arrays: Example 1

Given a char array `s`, where

`s = 'hello'`

We'd like to reverse it to obtain

`r = 'olleh'`

How to we write code to do this?

Solution:

Character of index `c` should be exchanged with character of index $(nc - c) + 1$.

Character and char arrays: Example 1

Given a char array s, where

s = 'hello'

We'd like to reverse it to obtain

r = 'olleh'

How to we write code to do this?

Solution:

Character of index c should be exchanged with character of index $(nc - c) + 1$.

```
n = length(s);
```

```
for k = 1:n
```

```
    r(k) = s(n-k+1);
```

```
end
```


Character and char arrays: Example 2

Given a char array `s` and a character `c`, we'd like to display the number of times `c` occurs in `s`.

Examples:

`s = 'mathematics', c = 'a' → display 2.`

How to we write code to do this?

Character and char arrays: Example 2

Given a char array `s` and a character `c`, we'd like to display the number of times `c` occurs in `s`.

Examples:

`s = 'mathematics', c = 'a' → display 2.`

How to we write code to do this?

Solution:

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

```
end
```

Character and char arrays: Example 2

Given a char array `s` and a character `c`, we'd like to display the number of times `c` occurs in `s`.

Examples:

`s = 'mathematics', c = 'a' → display 2.`

How to we write code to do this?

Solution:

```
for k = 1:length(s)
    if strcmp(s(k), c)

    end
end
```

Character and char arrays: Example 2

Given a char array `s` and a character `c`, we'd like to display the number of times `c` occurs in `s`.

Examples:

`s = 'mathematics', c = 'a' → display 2.`

How to we write code to do this?

Solution:

```
count = 0;
for k = 1:length(s)
    if strcmp(s(k), c)
        count = count+1;
    end
end
disp(count)
```

Cell arrays

Arrays (e.g. vectors, matrices, 3-D arrays, etc.)

- Can hold *one scalar* value in each of its components, e.g. one double, one char, one uint8.
- Data of all components must be the same type

Cell arrays

Arrays (e.g. vectors, matrices, 3-D arrays, etc.)

- Can hold *one scalar* value in each of its components, e.g. one double, one char, one uint8.
- Data of all components must be the same type

Cell arrays

- Each cell can store something “larger” than a scalar (but doesn’t have to).
Can store a vector in a single component, or a matrix, or a string, etc.
- Each cell can store something of a different type

Cell arrays: useful commands

Initialize a cell array with cell(...) function	<pre>c = cell(1,3); % Cell array with 1 row, 3 columns</pre>
Obtain number of rows and columns	<pre>[nr, nc] = size(c); % Same as for other arrays</pre>
Put items (char arrays, in this case) into the cell array	<pre>c = {'matlab', 'is', 'fun'}; % Commas optional</pre>
Display first item in cell 1 (which is a char array)	<pre>disp(c{1}) % Note the use of curly braces</pre>
Display first two cells	<pre>disp(c(1:2)) % Note the use of parentheses</pre>
Display first three elements of first cell	<pre>disp(c{1}(1:3)) % Note the use of curly braces and parentheses</pre>
Concatenate the char arrays (produces 'matlab is fun')	<pre>s = [c{1} ' ' c{2} ' ' c{3}] % Note the use of square brackets to create a string</pre>

Linear Search

Linear search is an algorithm for finding an element within an array.

Linear Search

Linear search is an algorithm for finding an element within an array.

For example, find if x is in vec

```
k = 1;
```

```
while k <= length(vec) && vec(k) ~= x
```

```
    k = k + 1;
```


```
end
```

Linear Search

Linear search is an algorithm for finding an element within an array.

For example, find if x is in vec

Check that the index
is not out of bounds



```
k = 1;
```

```
while k <= length(vec) && vec(k) ~= x
```

```
    k = k + 1;
```

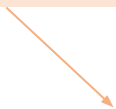
```
end
```

Linear Search

Linear search is an algorithm for finding an element within an array.

For example, find if x is in vec

Check if you have
found the element you
were looking for



$k = 1;$

while $k \leq \text{length}(vec) \ \&\& \ vec(k) \neq x$

$k = k + 1;$



end

If not, advance to the
next element

Linear Search

Linear search is an algorithm for finding an element within an array.

For example, find if x is in vec

$k = 1;$

while $k \leq \text{length}(vec) \ \&\& \ vec(k) \neq x$

$k = k + 1;$

end

if $k > \text{length}(vec)$

$found = false;$

else

$found = true;$

end

If the while loop is stopped because of k , it means element is not found,

else, it means element is found.

Linear Search

Linear search is an algorithm for finding an element within an array.

For example, find if x is in `vec`

```
k = 1;
```

```
while k <= length(vec) && vec(k) ~= x
```

```
    k = k + 1;
```

```
end
```

```
if k > length(vec)
```

```
    found = false;
```

```
else
```

```
    found = true;
```

```
end
```

Note: Using a *while loop* is more efficient than a *for loop* in this case since it allows you to stop looking after you have found the element you were looking for

Practice Problem

A 2D cell array S stores email address, for example

$S = \{ \text{'ste652@cornell.edu'}, \text{'egf34@cornell.edu'}, \text{'jkl179@cornell.edu'}, \dots$
 $\text{'ab4@cornell.edu'}, \text{'sn8@cornell.edu'}, \text{'bs435@cornell.edu'}, \dots \},$

Find all the netID that contains the letter a in it.

Practice Problem

A 2D cell array S stores email address, for example

$S = \{ \text{'ste652@cornell.edu'}, \text{'egf34@cornell.edu'}, \text{'jkl179@cornell.edu'}, \dots$
 $\text{'ab4@cornell.edu'}, \text{'sn8@cornell.edu'}, \text{'bs435@cornell.edu'}, \dots \},$

Generate a 2D cell array T with the same size as S , which only contains the netIDs. Keep the same order.

Decomposition:

- Loop over all email addresses.

Practice Problem

A 2D cell array S stores email address, for example

$S = \{ \text{'ste652@cornell.edu'}, \text{'egf34@cornell.edu'}, \text{'jkl179@cornell.edu'}, \dots$
 $\text{'ab4@cornell.edu'}, \text{'sn8@cornell.edu'}, \text{'bs435@cornell.edu'}, \dots \},$

Generate a 2D cell array T with the same size as S , which only contains the netIDs. Keep the same order.

Decomposition:

- Loop over all email addresses.
- For each email address, find the netID

Practice Problem

A 2D cell array S stores email address, for example

$S = \{ \text{'ste652@cornell.edu'}, \text{'egf34@cornell.edu'}, \text{'jkl179@cornell.edu'}, \dots$
 $\text{'ab4@cornell.edu'}, \text{'sn8@cornell.edu'}, \text{'bs435@cornell.edu'}, \dots \},$

Generate a 2D cell array T with the same size as S , which only contains the netIDs. Keep the same order.

Decomposition:

- Loop over all email addresses.
- For each email address, find the netID \rightarrow Linear search for the character @

Practice Problem

A 2D cell array S stores email address, for example

$S = \{ \text{'ste652@cornell.edu'}, \text{'egf34@cornell.edu'}, \text{'jkl179@cornell.edu'}, \dots$
 $\text{'ab4@cornell.edu'}, \text{'sn8@cornell.edu'}, \text{'bs435@cornell.edu'}, \dots \},$

Generate a 2D cell array T with the same size as S , which only contains the netIDs. Keep the same order.

Decomposition:

- Loop over all email addresses.
- For each email address, find the netID \rightarrow Linear search for the character @
- store each netID into the corresponding position of T

$S = \{ \text{'ste652@cornell.edu'}, \text{'egf34@cornell.edu'}, \text{'jkl179@cornell.edu'}, \dots$
 $\text{'ab4@cornell.edu'}, \text{'sn8@cornell.edu'}, \text{'bs435@cornell.edu'}, \dots \},$

Solution:

```
[m, n] = size(S); T = cell(m,n)
```

```
for i = 1:n
```

```
    for j = 1:m
```

```
        email = S{i,j}; netID = "";
```

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

```
            if email(k) ~= '@'
```

```
                netID = [netID email(k)];
```

```
            end
```

```
        end
```

```
        T{i,j} = netID
```

```
    end
```

```
end
```

Important functions in processing a file

Read a file

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

Test end-of-file

```
feof(fid);
```

Read a line (returns the next line of the specified file)

```
fgetl(fid);
```

Close a file

```
fclose(fid);
```

Q&A

- Slides and the recording will be posted.
- More questions attached at the end.
- Good luck!

Fall 2016 Prelim: Question 5a

Question 5a: (10 points)

Implement the following function as specified:

```
function v = getIndices(str, sep)
% str and sep are each a string and str is longer than sep. sep is the
% separator string, i.e., the delimiting string.
% v is the vector of the indices where the separator begins in str.
% Therefore the length of v is the number of times that sep occurs in str.
% Examples: If str is 'Hi!?Ann!?Bob' and sep is '!?' then v is [3 8].
%           If str is 'Hi!Ann!Bob' and sep is '!' then v is [3 7].
%           If str is 'Hi!Ann!Bob' and sep is '?' then v is [].
% Assume that the characters in sep are used only as the delimiter and not
% in the separated substrings. Assume that separators are always correctly
% placed--never incomplete and never side-by-side not separating anything.
% DO NOT USE any built-in functions other than length and strcmp.
```

Fall 2016 Prelim: Question 5a

Designing an algorithm

#	Thing we need to do	Programming concept needed to do this thing
1	Loop through all substrings of str with the same length as sep	
2		
3		

Fall 2016 Prelim: Question 5a

Designing an algorithm

#	Thing we need to do	Programming concept needed to do this thing
1	Loop through all substrings of str with the same length as sep	A loop on the index of the leftmost character of the substring for left=1:length(str)-length(sep)+1 (why?)
2		
3		

Fall 2016 Prelim: Question 5a

Designing an algorithm

#	Thing we need to do	Programming concept needed to do this thing
1	Loop through all substrings of str with the same length as sep	A loop on the index of the leftmost character of the substring for left=1:length(str)-length(sep)+1 (why?)
2	For a given left index, extract substring with same length as sep and compare with sep	
3		

Fall 2016 Prelim: Question 5a

Designing an algorithm

#	Thing we need to do	Programming concept needed to do this thing
1	Loop through all substrings of str with the same length as sep	A loop on the index of the leftmost character of the substring for left=1:length(str)-length(sep)+1 (why?)
2	For a given left index, extract substring with same length as sep and compare with sep	substr = str(left:left+length(sep)-1); (why?) strcmp(substr,sep);
3		

Fall 2016 Prelim: Question 5a

Designing an algorithm

#	Thing we need to do	Programming concept needed to do this thing
1	Loop through all substrings of str with the same length as sep	A loop on the index of the leftmost character of the substring for left=1:length(str)-length(sep)+1 (why?)
2	For a given left index, extract substring with same length as sep and compare with sep	<code>substr = str(left:left+length(sep)-1);</code> (why?) <code>strcmp(substr,sep);</code>
3	Record the left index to the output variable whenever the substring matches sep	

Fall 2016 Prelim: Question 5a

Designing an algorithm

#	Thing we need to do	Programming concept needed to do this thing
1	Loop through all substrings of str with the same length as sep	A loop on the index of the leftmost character of the substring for left=1:length(str)-length(sep)+1 (why?)
2	For a given left index, extract substring with same length as sep and compare with sep	<code>substr = str(left:left+length(sep)-1);</code> (why?) <code>strcmp(substr,sep);</code>
3	Record the left index to the output variable whenever the substring matches sep	Append the left to v each time the above condition is true

Fall 2016 Prelim: Question 5a

Translating what we need to do into code:

```
for left = 1 : length(str) – length(sep) + 1
```

```
end
```

Connection to previous slide:

Red: for-loop on left index

Green: use strcmp on substr

Blue: append left index to v

Fall 2016 Prelim: Question 5a

Translating what we need to do into code:

```
for left = 1 : length(str) - length(sep) + 1
    substr = str(left : left + length(sep) - 1);
    if strcmp( substr, sep ) == 1

end
end
```

Connection to previous slide:

Red: for-loop on left index

Green: use strcmp on substr

Blue: append left index to v

Fall 2016 Prelim: Question 5a

Translating what we need to do into code:

```
v = [];  
for left = 1 : length(str) - length(sep) + 1  
    substr = str(left : left + length(sep) - 1);  
    if strcmp( substr, sep ) == 1  
        v = [v, left];  
    end  
end
```

Connection to previous slide:

Red: for-loop on left index

Green: use strcmp on substr

Blue: append left index to v

Fall 2016 Prelim: Question 5b

Question 5b: (25 points)

Assume that function `getIndices` of Question 5a has been correctly implemented; make effective use of it in implementing function `aveScores` below. Note the example at the bottom of the page.

```
function CA = aveScores(M)
% M is a 2-d array of characters. Each row of M stores the scores of one student:
%   a netID followed by one or more scores and these data items are separated by
%   commas. There may be trailing spaces in a row of M.
% CA is an n-by-2 cell array where n is the number of students whose record includes
%   at least two scores. In each row of CA, the first cell stores the netID of a
%   student who has at least two scores and the second cell stores the average score
%   of that student. If no student has at least two scores then CA is an empty cell
%   array.
% ONLY these built-in functions are allowed: length, size, str2double, sum, mean
% Recall that str2double can handle leading and trailing spaces, e.g.,
%   str2double('87   ') returns the type double scalar 87.
```

Example: Suppose M is

```
['vaf34,80,100,90';...
 'aaj91,100      ';...
 'rt2253,75,95   ']
```

Then `aveScores(M)` should return a 2×2 cell array CA:

- In row 1 column 1 is 'vaf34' and in row 1 column 2 is the type double scalar 90.
- In row 2 column 1 is 'rt2253' and in row 2 column 2 is the type double scalar 85.

Fall 2016 Prelim: Question 5b

Designing an algorithm

#	Thing we need to do	Programming concept needed to do this thing
1	Loop through the individual strings (rows) of M	
2		
3		
4		
5		

Fall 2016 Prelim: Question 5b

Designing an algorithm

#	Thing we need to do	Programming concept needed to do this thing
1	Loop through the individual strings (rows) of M	A for-loop that iterates for row_M = 1:size(M,1)
2		
3		
4		
5		

Fall 2016 Prelim: Question 5b

Designing an algorithm

#	Thing we need to do	Programming concept needed to do this thing
1	Loop through the individual strings (rows) of M	A for-loop that iterates for row_M = 1:size(M,1)
2	Find the commas in a given string. Skip to the next row if there is less than 2 test scores	
3		
4		
5		

Fall 2016 Prelim: Question 5b

Designing an algorithm

#	Thing we need to do	Programming concept needed to do this thing
1	Loop through the individual strings (rows) of M	A for-loop that iterates for row_M = 1:size(M,1)
2	Find the commas in a given string. Skip to the next row if there is less than 2 test scores	Use the getIndices function from the part 5a to find the indices of the commas comma_idx . Use an if statement to check if the string has at least 2 scores
3		
4		
5		

Fall 2016 Prelim: Question 5b

Designing an algorithm

#	Thing we need to do	Programming concept needed to do this thing
1	Loop through the individual strings (rows) of M	A for-loop that iterates for row_M = 1:size(M,1)
2	Find the commas in a given string. Skip to the next row if there is less than 2 test scores	Use the getIndices function from the part 5a to find the indices of the commas comma_idx . Use an if statement to check if the string has at least 2 scores
3	If there are at least two test scores in the row, extract the netID and store it in the first column of CA	
4		
5		

Fall 2016 Prelim: Question 5b

Designing an algorithm

#	Thing we need to do	Programming concept needed to do this thing
1	Loop through the individual strings (rows) of M	A for-loop that iterates for row_M = 1:size(M,1)
2	Find the commas in a given string. Skip to the next row if there is less than 2 test scores	Use the getIndices function from the part 5a to find the indices of the commas comma_idx . Use an if statement to check if the string has at least 2 scores
3	If there are at least two test scores in the row, extract the netID and store it in the first column of CA	Since not all rows of M will be stored in output CA, set up a rowCA index which updates each step. Then CA{row_CA, 1} = M(rowM , 1:comma_idx(1)-1);
4		
5		

Fall 2016 Prelim: Question 5b

Designing an algorithm

#	Thing we need to do	Programming concept needed to do this thing
1	Loop through the individual strings (rows) of M	A for-loop that iterates for row_M = 1:size(M,1)
2	Find the commas in a given string. Skip to the next row if there is less than 2 test scores	Use the getIndices function from the part 5a to find the indices of the commas comma_idx . Use an if statement to check if the string has at least 2 scores
3	If there are at least two test scores in the row, extract the netID and store it in the first column of CA	Since not all rows of M will be stored in output CA, set up a rowCA index which updates each step. Then CA{row_CA, 1} = M(rowM , 1:comma_idx(1)-1);
4	Knowing the indices of the commas, loop through the corresponding substrings to extract test scores	
5		

Fall 2016 Prelim: Question 5b

Designing an algorithm

#	Thing we need to do	Programming concept needed to do this thing
1	Loop through the individual strings (rows) of M	A for-loop that iterates for row_M = 1:size(M,1)
2	Find the commas in a given string. Skip to the next row if there is less than 2 test scores	Use the getIndices function from the part 5a to find the indices of the commas comma_idx . Use an if statement to check if the string has at least 2 scores
3	If there are at least two test scores in the row, extract the netID and store it in the first column of CA	Since not all rows of M will be stored in output CA, set up a rowCA index which updates each step. Then CA{row_CA, 1} = M(rowM , 1:comma_idx(1)-1);
4	Knowing the indices of the commas, loop through the corresponding substrings to extract test scores	Use another for-loop (nested inside the first) that iterates from k = 1:length(comma_idx) , and determine indices of substring
5		

Fall 2016 Prelim: Question 5b

Designing an algorithm

#	Thing we need to do	Programming concept needed to do this thing
1	Loop through the individual strings (rows) of M	A for-loop that iterates for row_M = 1:size(M,1)
2	Find the commas in a given string. Skip to the next row if there is less than 2 test scores	Use the getIndices function from the part 5a to find the indices of the commas comma_idx . Use an if statement to check if the string has at least 2 scores
3	If there are at least two test scores in the row, extract the netID and store it in the first column of CA	Since not all rows of M will be stored in output CA, set up a rowCA index which updates each step. Then CA{row_CA, 1} = M(rowM , 1:comma_idx(1)-1);
4	Knowing the indices of the commas, loop through the corresponding substrings to extract test scores	Use another for-loop (nested inside the first) that iterates from k = 1:length(comma_idx) , and determine indices of substring
5	Store as a running sum in the second column of CA, and take the average after all scores have been extracted.	

Fall 2016 Prelim: Question 5b

Designing an algorithm

#	Thing we need to do	Programming concept needed to do this thing
1	Loop through the individual strings (rows) of M	A for-loop that iterates for row_M = 1:size(M,1)
2	Find the commas in a given string. Skip to the next row if there is less than 2 test scores	Use the getIndices function from the part 5a to find the indices of the commas comma_idx . Use an if statement to check if the string has at least 2 scores
3	If there are at least two test scores in the row, extract the netID and store it in the first column of CA	Since not all rows of M will be stored in output CA, set up a rowCA index which updates each step. Then CA{row_CA, 1} = M(rowM , 1:comma_idx(1)-1);
4	Knowing the indices of the commas, loop through the corresponding substrings to extract test scores	Use another for-loop (nested inside the first) that iterates from k = 1:length(comma_idx) , and determine indices of substring
5	Store as a running sum in the second column of CA, and take the average after all scores have been extracted.	Initialize the second column to zero outside the for-loop of step 4, then convert the substring to a double and add to the second column.

Fall 2016 Prelim: Question 5b

```
for rowM = 1:size(M,1)
```

Connection to previous slide:

Red: for-loop to look at each string in M

Orange: extract commas

Green: extract and store netID

Blue: extract test score indices

Black: compute average test score

```
end
```

Fall 2016 Prelim: Question 5b

```
for rowM = 1:size(M,1)
    comma_idx = getIndices( M( rowM , : ), ',' );
    if length(comma_idx) >= 2

        netID = M( rowM , comma_idx(1) );
        testScoreIndices = M( rowM , comma_idx(2) );
        avgTestScore = mean( M( rowM , testScoreIndices ) );

    end
end
```

Connection to previous slide:

Red: for-loop to look at each string in M

Orange: extract commas

Green: extract and store netID

Blue: extract test score indices

Black: compute average test score

Fall 2016 Prelim: Question 5b

```
rowCA = 0;
for rowM = 1:size(M,1)
    comma_idx = getIndices( M( rowM , : ), ',' );
    if length(comma_idx) >= 2
        rowCA = rowCA+1;
        CA{ rowCA, 1 } = M( rowM, 1:comma_idx(1)-1 );
    end
end
```

Connection to previous slide:

Red: for-loop to look at each string in M

Orange: extract commas

Green: extract and store netID

Blue: extract test score indices

Black: compute average test score

Fall 2016 Prelim: Question 5b

```
rowCA = 0;
for rowM = 1:size(M,1)
    comma_idx = getIndices( M( rowM , : ), ',' );
    if length(comma_idx) >= 2
        rowCA = rowCA+1;
        CA{ rowCA, 1 } = M( rowM, 1:comma_idx(1)-1 );

        for k = 1:length(comma_idx)
            left = comma_idx(k)+1;
            if k < length(comma_idx)
                right = comma_idx(k+1)-1;
            else
                right = size(M,2); % After last comma, take all remaining characters
            end
        end
    end
end
end
end
```

Connection to previous slide:

Red: for-loop to look at each string in M

Orange: extract commas

Green: extract and store netID

Blue: extract test score indices

Black: compute average test score

Fall 2016 Prelim: Question 5b

```
rowCA = 0; CA = {};
for rowM = 1:size(M,1)
    comma_idx = getIndices( M( rowM , : ), ',' );
    if length(comma_idx) >= 2
        rowCA = rowCA+1;
        CA{ rowCA, 1 } = M( rowM, 1:comma_idx(1)-1 );
        CA{ rowCA, 2 } = 0;
        for k = 1:length(comma_idx)
            left = comma_idx(k)+1;
            if k < length(comma_idx)
                right = comma_idx(k+1)-1;
            else
                right = size(M,2); % After last comma, take all remaining characters
            end
            CA{rowCA,2} = CA{rowCA,2} + str2double( M( rowM, left:right ) );
        end
        CA{rowCA,2} = CA{rowCA,2}/length(comma_idx);
    end
end
```

Connection to previous slide:

Red: for-loop to look at each string in M

Orange: extract commas

Green: extract and store netID

Blue: extract test score indices

Black: compute average test score

Fall 2018 Prelim: Question 5

Question 5: (25 points)

Implement the following function as specified.

```
function counts = wordBag(text, vocab)
% Determine how many times each word of a vocabulary appears in a sentence.
% text: The sentence. A period terminated char row vector containing words in lower-
% case. Words are separated by a single space and there is no punctuation mark other
% than the period at the end.
% vocab: A 1-d cell array of distinct words (char row vectors) representing a vocabulary
% counts: A type double vector the same length as vocab storing the number of times
% each word of vocab appears in text.
% Example: Suppose text= 'the red stone and red walls.'
%           and      vocab= {'red', 'blue', 'wall', 'stone'}
%           Then      wordBag(text, vocab)
%           returns    [2 0 0 1]
%           because the first word of vocab appears twice in text, the second word of vocab
%           appears 0 times in text, ..., and so forth.
% The only built-in functions allowed are zeros, cell, length, size, strcmp
```


Fall 2018 Prelim: Question 5

Designing an algorithm

#	Thing we need to do	Programming concept needed to do this thing
1	Look through all of text	
2		
3		
4		

Fall 2018 Prelim: Question 5

Designing an algorithm

#	Thing we need to do	Programming concept needed to do this thing
1	Look through all of text	A for-loop that iterates for k = 1:length(text)
2		
3		
4		

Fall 2018 Prelim: Question 5

Designing an algorithm

#	Thing we need to do	Programming concept needed to do this thing
1	Look through all of text	A for-loop that iterates for k = 1:length(text)
2	Find each word in text	
3		
4		

Fall 2018 Prelim: Question 5

Designing an algorithm

#	Thing we need to do	Programming concept needed to do this thing
1	Look through all of text	A for-loop that iterates for k = 1:length(text)
2	Find each word in text	An if statement that checks if the a word ending has not been reached: if text(k)~=' ' && text(k)~='.' and builds onto the current word if this is true: curWord= [curWord text(k)];
3		
4		

Fall 2018 Prelim: Question 5

Designing an algorithm

#	Thing we need to do	Programming concept needed to do this thing
1	Look through all of text	A for-loop that iterates for k = 1:length(text)
2	Find each word in text	An if statement that checks if the a word ending has not been reached: if text(k)~=' ' && text(k)~='.' and builds onto the current word if this is true: curWord= [curWord text(k)];
3	Search for each completed word in vocab	
4		

Fall 2018 Prelim: Question 5

Designing an algorithm

#	Thing we need to do	Programming concept needed to do this thing
1	Look through all of text	A for-loop that iterates for k = 1:length(text)
2	Find each word in text	An if statement that checks if the a word ending has not been reached: if text(k)~= ' ' && text(k)~='.' and builds onto the current word if this is true: curWord= [curWord text(k)];
3	Search for each completed word in vocab	Use linear search to find the first occurrence of the current word in the cell array vocab
4		

Fall 2018 Prelim: Question 5

Designing an algorithm

#	Thing we need to do	Programming concept needed to do this thing
1	Look through all of text	A for-loop that iterates for k = 1:length(text)
2	Find each word in text	An if statement that checks if the a word ending has not been reached: if text(k)~= ' ' && text(k)~='.' and builds onto the current word if this is true: curWord= [curWord text(k)];
3	Search for each completed word in vocab	Use linear search to find the first occurrence of the current word in the cell array vocab
4	Update the number of times that the word has occurred	

Fall 2018 Prelim: Question 5

Designing an algorithm

#	Thing we need to do	Programming concept needed to do this thing
1	Look through all of text	A for-loop that iterates for k = 1:length(text)
2	Find each word in text	An if statement that checks if the a word ending has not been reached: if text(k)~= ' ' && text(k)~='.' and builds onto the current word if this is true: curWord= [curWord text(k)];
3	Search for each completed word in vocab	Use linear search to find the first occurrence of the current word in the cell array vocab
4	Update the number of times that the word has occurred	Increment the correct index of counts if the current word exists in vocab

Fall 2018 Prelim: Question 5

for k= 1:length(text)

end

Connection to previous slide:

Red: for-loop to look at each char in text

Orange: build a word

Green: search for a word in vocab

Blue: update the count

Fall 2018 Prelim: Question 5

```
curWord= '';  
for k= 1:length(text)  
    if text(k)~=' ' && text(k)~='.'  
        curWord= [curWord text(k)];  
    else
```

```
        curWord= '';  
    end  
end
```

Connection to previous slide:

Red: for-loop to look at each char in text

Orange: build a word

Green: search for a word in vocab


Blue: update the count

Fall 2018 Prelim: Question 5

```
curWord= '';  
for k= 1:length(text)  
    if text(k)~=' ' && text(k)~='.'  
        curWord= [curWord text(k)];  
    else
```

```
        curWord= '';  
    end  
end
```

Reset curWord after
the end of a word is
reached



Connection to previous slide:

Red: for-loop to look at each char in text

Orange: build a word

Green: search for a word in vocab

Blue: update the count

Fall 2018 Prelim: Question 5

```
nv= length(vocab);
```

```
curWord= '';
```

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

```
    if text(k)~=' ' && text(k)~='.'
```

```
        curWord= [curWord text(k)];
```

```
    else
```

```
        j= 1;
```

```
        while j<=nv && ~strcmp(vocab{j}, curWord)
```

```
            j= j + 1;
```

```
        end
```

```
        curWord= '';
```

```
    end
```

```
end
```

Connection to previous slide:

Red: for-loop to look at each char in text

Orange: build a word

Green: search for a word in vocab

Blue: update the count

Fall 2018 Prelim: Question 5

```
nv= length(vocab);
counts= zeros(1,nv);
curWord= "";
for k= 1:length(text)
    if text(k)~=' ' && text(k)~='.'
        curWord= [curWord text(k)];
    else
        j= 1;
        while j<=nv && ~strcmp(vocab{j}, curWord)
            j= j + 1;
        end
        if j<=nv
            counts(j)= counts(j) + 1;
        end
        curWord= "";
    end
end
```

Connection to previous slide:

Red: for-loop to look at each char in text



Orange: build a word

Green: search for a word in vocab

Blue: update the count

Common Student Errors

- Getting the size of an array/Initializing arrays

`size(A) = [nr, nc];`  **vs** `[nr, nc] = size(A);` 

- For loops based on array size

`for k = 1:length(nr)`  **vs** `for k = 1:nr` 

- 2D Cell Array vs. Arrays in Cells

`A{1, 2}`

`A{1}(2)`

`{1, 2, 3;
4, 5, 6}`

`{[1, 2, 3],
[4, 6]}`

has 6 cells

has 2 cells