- Previous Lecture:
  - Working with images
- Today's Lecture:
  - Characters and strings
- Announcements:
  - Prelim 2 will be returned at end of lecture. If your paper isn't here, pick it up from CS1112 consultants in ACCEL during consulting hrs (starting today after 4pm)
  - Discussion this week in classrooms as listed on roster
  - Project 4 posted. Due Mon, Nov 1st, at 11pm

Characters & strings

- We have used strings already:
  - `n= input('Next number: ')`
  - `sprintf('Answer is %d', ans)`
- A string is made up of individual characters, so a string is a 1-d array of characters
- '`CS1112 rocks!`' is a character array of length 13; it has 7 letters, 4 digits, 1 space, and 1 symbol.

'C'	'S'	'1'	'1'	'1'	'2'	' '	'r'	'o'	'c'	'k'	's'	'!'

Strings are important in computation

Numerical data is often encoded in strings. E.g., a file containing Ithaca weather data begins with the string

W07629N4226

meaning

Longitude:	76° 29' West
Latitude:	42° 26' North

We may need to grab hold of the substring **W07629**, convert **076** and **29** to the numeric values 76 and 29, and do some computation

Comparison of genomic sequences is another example of string computation

- E.g., looking for a pattern:

  Given the sequence **ATTCTGACCTCGATC…**

  Look for the pattern **ACCT**
- E.g., quantifying the difference between sequences:

  ATTCTGACCTCGATC

  ATTCGTGACCTCGAT

What if this nucleotide is removed?

Single quotes enclose strings in Matlab

Anything enclosed in single quotes is a string (*even if it looks like something else*)

- `'100'` is a character array (string) of length 3
- `100` is a numeric value
- `'pi'` is a character array of length 2
- `pi` is the built-in constant 3.1416…
- `'x'` is a character (vector of length 1)
- `x` may be a variable name in your program

Strings as vectors

Vectors

- Assignment

  v= [7 0 5];
- Indexing

  x= v(3); % x is 5

  v(1)= 1; % v is [1 0 5]

  w= v(2:3); % w is [0 5]
- : notation

  v= 2:5; % v is [2 3 4 5]
- Appending

  v= [7 0 5];

  v(4)= 2; % v is [7 0 5 2]
- Concatenation

  v= [v [4 6]];

  % v is [7 0 5 2 4 6]

Strings

- Assignment

  s= 'hello';
- Indexing

  c= s(2); % c is 'e'

  s(1)= 'J'; % s is 'Jello'

  t= s(2:4); % t is 'ell'
- : notation

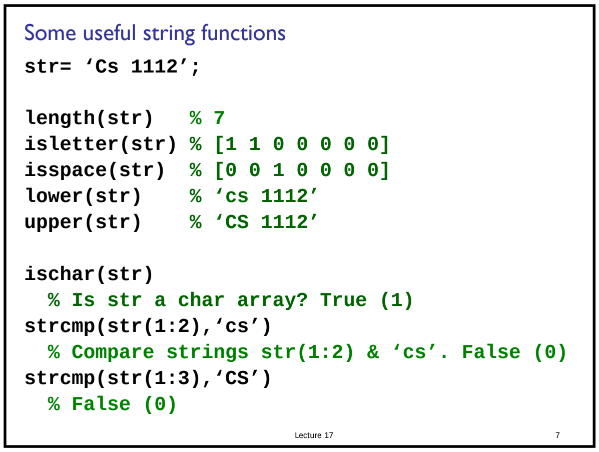
  s= 'a':'g'; % s is 'abcdefg'
- Appending

  s= 'duck';

  s(5)= 's'; % s is 'ducks'
- Concatenation

  s= [s ' quack'];

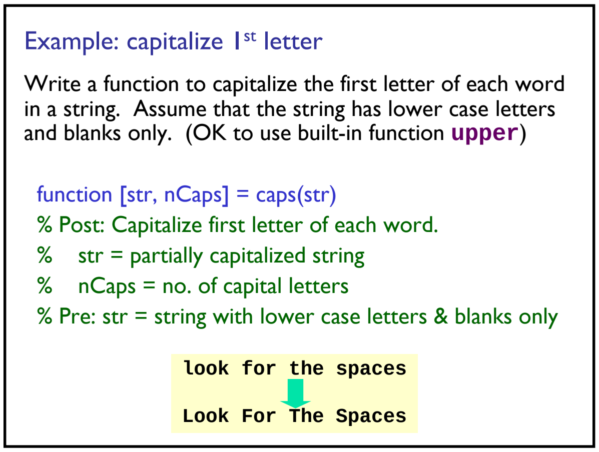
  % s is 'ducks quack'



Some useful string functions

```
str= ‘Cs 1112’;

length(str)   % 7
isletter(str) % [1 1 0 0 0 0 0]
isspace(str)  % [0 0 1 0 0 0 0]
lower(str)    % ‘cs 1112’
upper(str)    % ‘CS 1112’

ischar(str)
  % Is str a char array? True (1)
strcmp(str(1:2),‘cs’)
  % Compare strings str(1:2) & ‘cs’. False (0)
strcmp(str(1:3),‘CS’)
  % False (0)
```

Lecture 17 7

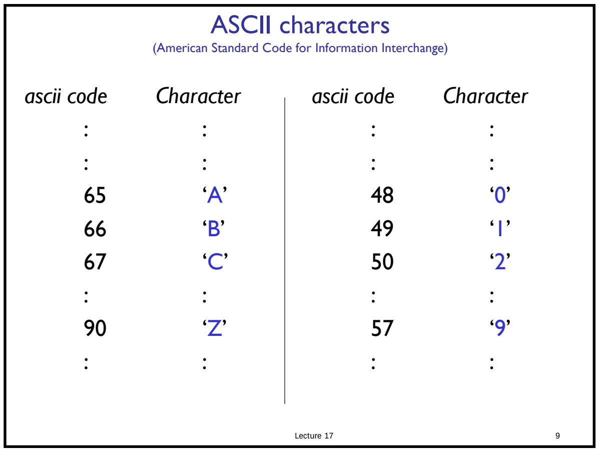


Example: capitalize 1st letter

Write a function to capitalize the first letter of each word in a string. Assume that the string has lower case letters and blanks only. (OK to use built-in function **upper**)

```
function [str, nCaps] = caps(str)
% Post: Capitalize first letter of each word.
%    str = partially capitalized string
%    nCaps = no. of capital letters
% Pre: str = string with lower case letters & blanks only
```

```
look for the spaces
Look For The Spaces
```

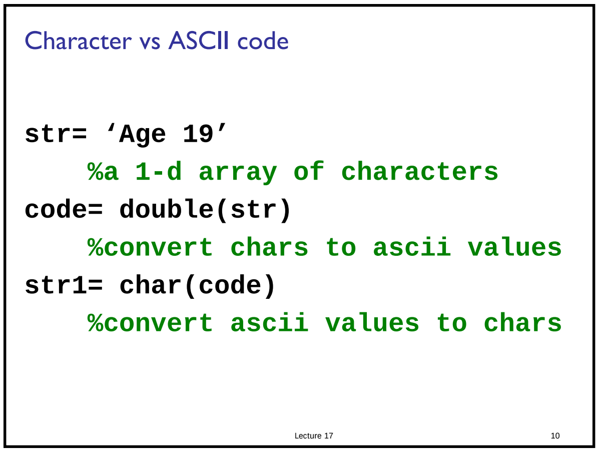


ASCII characters

(American Standard Code for Information Interchange)

ascii code	*Character*	*ascii code*	*Character*
:	:	:	:
:	:	:	:
65	‘A’	48	‘0’
66	‘B’	49	‘1’
67	‘C’	50	‘2’
:	:	:	:
90	‘Z’	57	‘9’
:	:	:	:

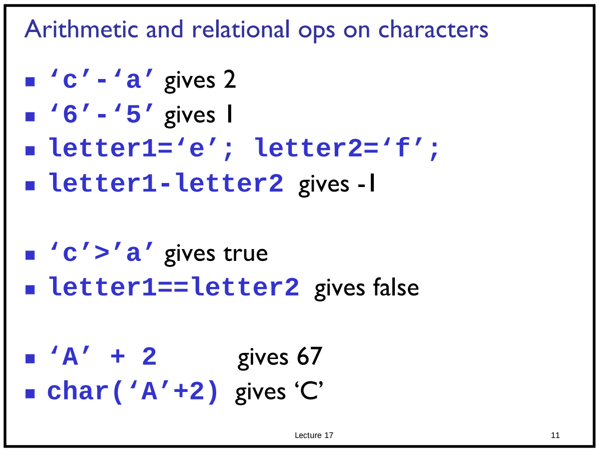
Lecture 17 9



Character vs ASCII code

```
str= ‘Age 19’
    %a 1-d array of characters
code= double(str)
    %convert chars to ascii values
str1= char(code)
    %convert ascii values to chars
```

Lecture 17 10



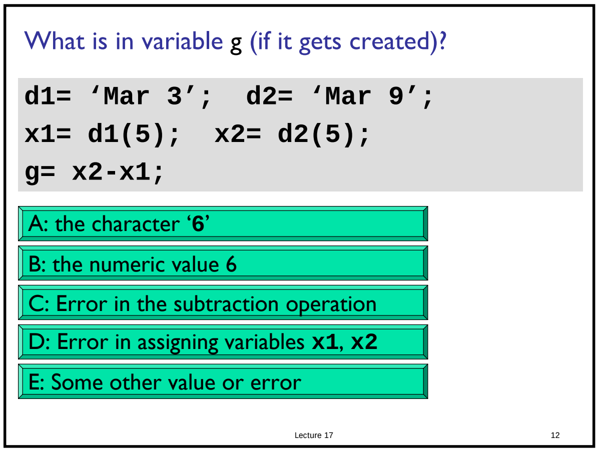
Arithmetic and relational ops on characters

- `‘c’-‘a’` gives 2
- `‘6’-‘5’` gives 1
- `letter1=‘e’; letter2=‘f’;`
- `letter1-letter2` gives -1

- `‘c’>’a’` gives true
- `letter1==letter2` gives false

- `‘A’ + 2` gives 67
- `char(‘A’+2)` gives ‘C’

Lecture 17 11



What is in variable g (if it gets created)?

```
d1= ‘Mar 3’;  d2= ‘Mar 9’;
x1= d1(5);  x2= d2(5);
g= x2-x1;
```

A: the character ‘**6**’

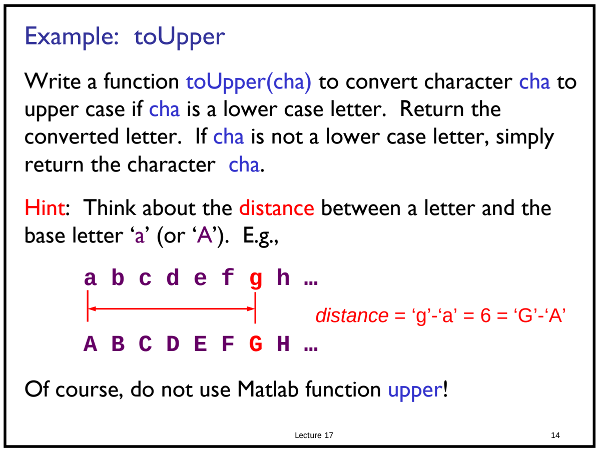
B: the numeric value 6

C: Error in the subtraction operation

D: Error in assigning variables **x1**, **x2**

E: Some other value or error

Lecture 17 12



Example: toUpper

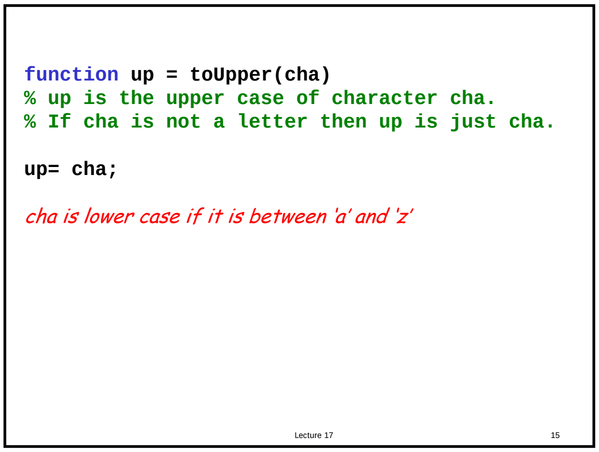
Write a function toUpper(cha) to convert character cha to upper case if cha is a lower case letter. Return the converted letter. If cha is not a lower case letter, simply return the character cha.

Hint: Think about the distance between a letter and the base letter 'a' (or 'A'). E.g.,

```
a b c d e f g h …

A B C D E F G H …
```

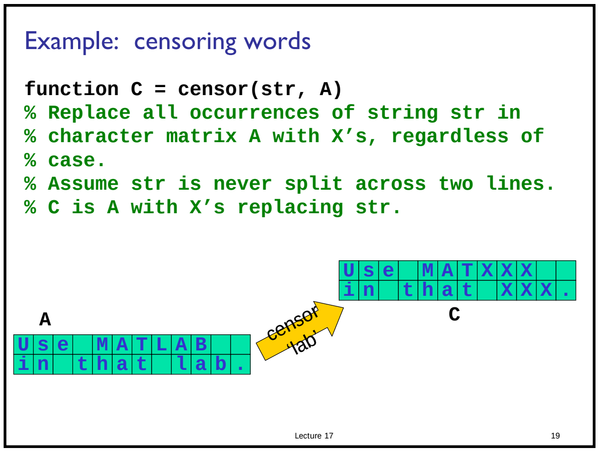
distance = 'g'-'a' = 6 = 'G'-'A'

Of course, do not use Matlab function upper!



```
function up = toUpper(cha)
% up is the upper case of character cha.
% If cha is not a letter then up is just cha.

up= cha;
```

cha is lower case if it is between 'a' and 'z'



Example: censoring words

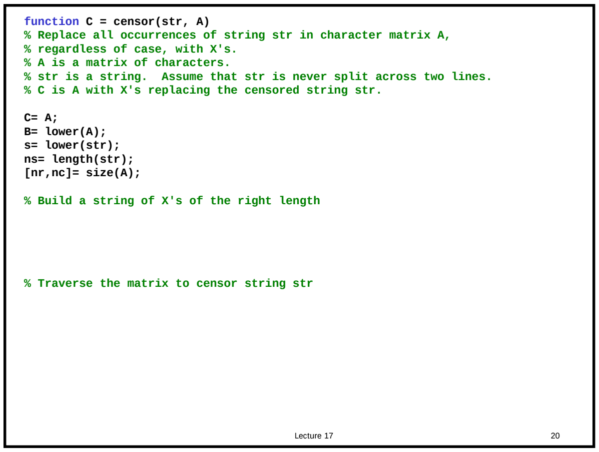
```
function C = censor(str, A)
% Replace all occurrences of string str in
% character matrix A with X's, regardless of
% case.
% Assume str is never split across two lines.
% C is A with X's replacing str.
```

A

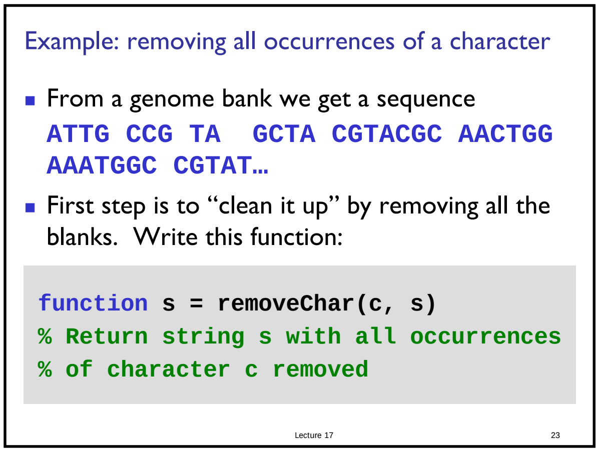
U	s	e		M	A	T	L	A	B		
i	n		t	h	a	t		l	a	b	.

C

U	s	e		M	A	T	X	X	X		
i	n		t	h	a	t		X	X	X	.



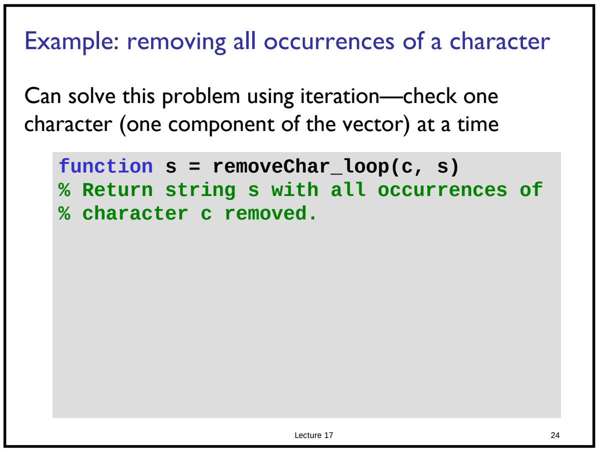
```
function C = censor(str, A)
% Replace all occurrences of string str in character matrix A,
% regardless of case, with X's.
% A is a matrix of characters.
% str is a string.  Assume that str is never split across two lines.
% C is A with X's replacing the censored string str.

C= A;
B= lower(A);
s= lower(str);
ns= length(str);
[nr,nc]= size(A);

% Build a string of X's of the right length




% Traverse the matrix to censor string str
```



Example: removing all occurrences of a character

- From a genome bank we get a sequence
  ATTG CCG TA GCTA CGTACGC AACTGG AAATGGC CGTAT…
- First step is to "clean it up" by removing all the blanks. Write this function:

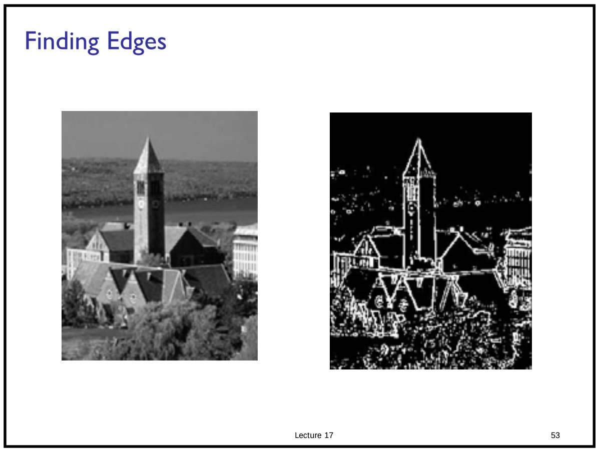
```
function s = removeChar(c, s)
% Return string s with all occurrences
% of character c removed
```



Example: removing all occurrences of a character

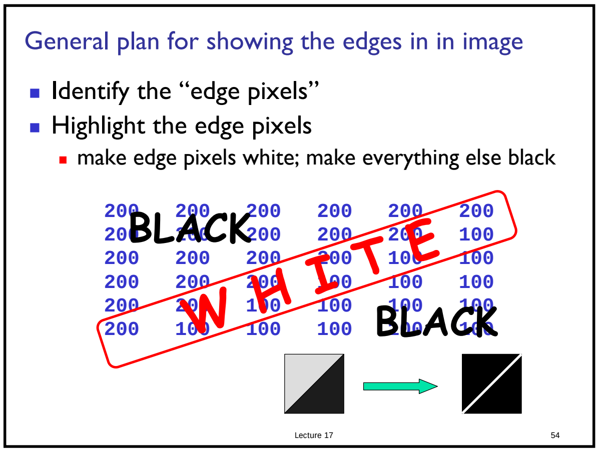
Can solve this problem using iteration—check one character (one component of the vector) at a time

```
function s = removeChar_loop(c, s)
% Return string s with all occurrences of
% character c removed.
```



Finding Edges

Lecture 17 53

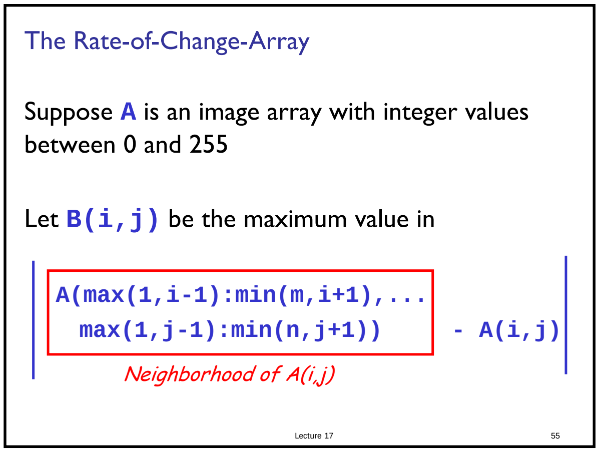


General plan for showing the edges in in image

- Identify the “edge pixels”
- Highlight the edge pixels
  - make edge pixels white; make everything else black

```
200  200  200  200  200  200
200  200  200  200  200  100
200  200  200  200  100  100
200  200  200  100  100  100
200  200  100  100  100  100
200  100  100  100  100  100
```

Lecture 17 54



The Rate-of-Change-Array

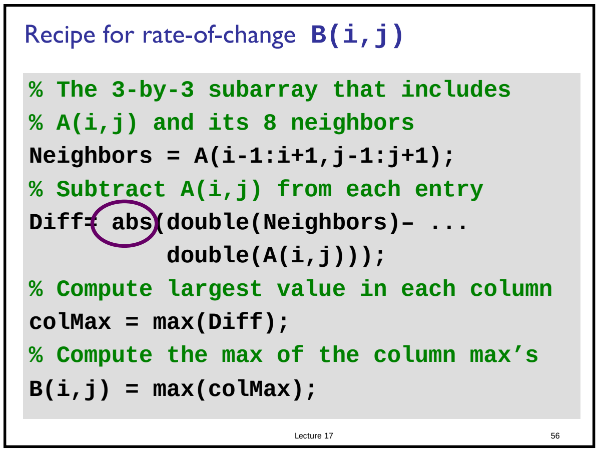
Suppose **A** is an image array with integer values between 0 and 255

Let **B(i,j)** be the maximum value in

```
| A(max(1,i-1):min(m,i+1),...          |
|   max(1,j-1):min(n,j+1))    - A(i,j) |
```

Neighborhood of A(i,j)

Lecture 17 55

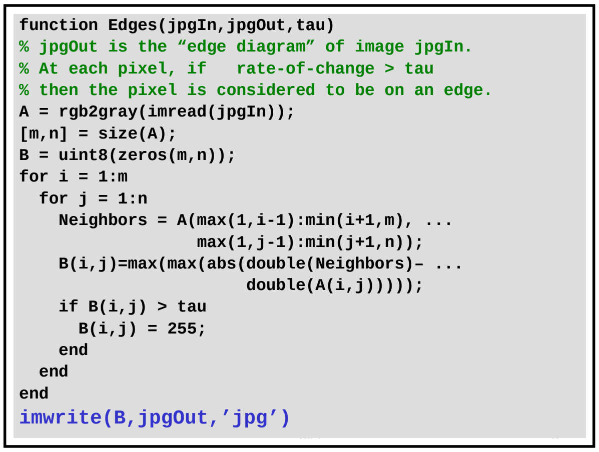


Recipe for rate-of-change B(i,j)

```
% The 3-by-3 subarray that includes
% A(i,j) and its 8 neighbors
Neighbors = A(i-1:i+1,j-1:j+1);
% Subtract A(i,j) from each entry
Diff= abs(double(Neighbors)- ...
            double(A(i,j)));
% Compute largest value in each column
colMax = max(Diff);
% Compute the max of the column max’s
B(i,j) = max(colMax);
```

Lecture 17 56

```
function Edges(jpgIn,jpgOut,tau)
% jpgOut is the “edge diagram” of image jpgIn.
% At each pixel, if   rate-of-change > tau
% then the pixel is considered to be on an edge.

A = rgb2gray(imread(jpgIn));
[m,n] = size(A);
B = uint8(zeros(m,n));
for i = 1:m
  for j = 1:n
    Neighbors = A(max(1,i-1):min(i+1,m), ...
                  max(1,j-1):min(j+1,n));
    B(i,j)=max(max(abs(double(Neighbors)- ...
                       double(A(i,j)))));



  end
end
```



```
function Edges(jpgIn,jpgOut,tau)
% jpgOut is the “edge diagram” of image jpgIn.
% At each pixel, if   rate-of-change > tau
% then the pixel is considered to be on an edge.
A = rgb2gray(imread(jpgIn));
[m,n] = size(A);
B = uint8(zeros(m,n));
for i = 1:m
  for j = 1:n
    Neighbors = A(max(1,i-1):min(i+1,m), ...
                  max(1,j-1):min(j+1,n));
    B(i,j)=max(max(abs(double(Neighbors)- ...
                       double(A(i,j)))));
    if B(i,j) > tau
      B(i,j) = 255;
    end
  end
end
imwrite(B,jpgOut,’jpg’)
```