- Today's Lecture:
  - Vectorized Coding
  - Review

- Announcements:
  - Thursday class at Stimson G25 computer lab.
  - Assignment 2 due Thursday 10/3 at 11:59pm
  - Test 2 Friday 10/4 at Upson B7

Vectorized Coding

```
function d = minDistance(x,y,z)
% Find the min distance between a set of points and the origin

nPoints = length(x);
d = zeros(nPoints,1);          % Preallocate

for k = 1:nPoints              % Compute distance for every point
   d(k) = sqrt(x(k)^2 + y(k)^2 + z(k)^2);
end

d = min(d);                    % Get the minimum distance
```

Vectorized Coding

```
function d = minDistance(x,y,z)
% Find the min distance between a set of points and the origin

nPoints = length(x);
d = zeros(nPoints,1);          % Preallocate

for k = 1:nPoints              % Compute distance for every point
   d(k) = sqrt(x(k)^2 + y(k)^2 + z(k)^2);
end

d = min(d);                    % Get the minimum distance
```

```
function d = minDistance(x,y,z)
% Find the min distance between a set of points and the origin

d = sqrt(x.^2 + y.^2 + z.^2);     % Compute distance for every point
d = min(d);                       % Get the minimum distance
```

Vectorized (logical) operations – 1d

```
a= [4 2 6 1 3];
b= [5 3 6 5 3];
Relational ops:
   L= a==b
   M= a>b
Arithmetic ops:
   c= a-b
Extraction:
   d= a(a==b)
   e= b(b>3 & rem(b,2)==1)
```

Vectorized (logical) operations – 2d

```
m= [ 2 3  5 7; …
    -2 1  0 7; …
     5 2 -1 8]

L= m>3

P= m>3 | m<0

a= m(m>4)

b= (m>4).*m
```

Examples

- Find all non-upper-case letters in str:

```
str(str<'A' | str>'Z')
```

- Replace these non-upper-case letters with a space:

```
str(str<'A' | str>'Z') = ' '
```

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’
:	:	:	:

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
```

Arithmetic and relational ops on characters

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

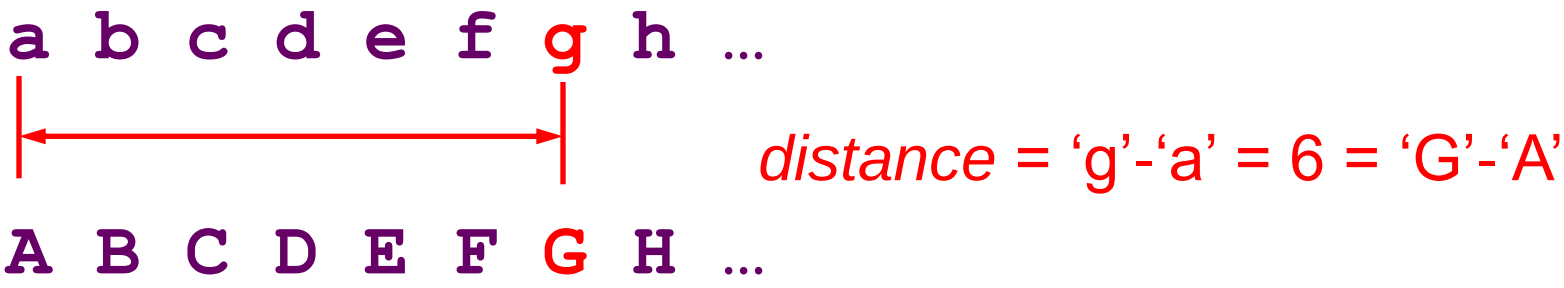
- `'c'>'a'` gives true
- `letter1==letter2` gives false

- `'A' + 2` gives 67
- `char('A'+2)` gives 'C'

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.,



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.
```

```
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'

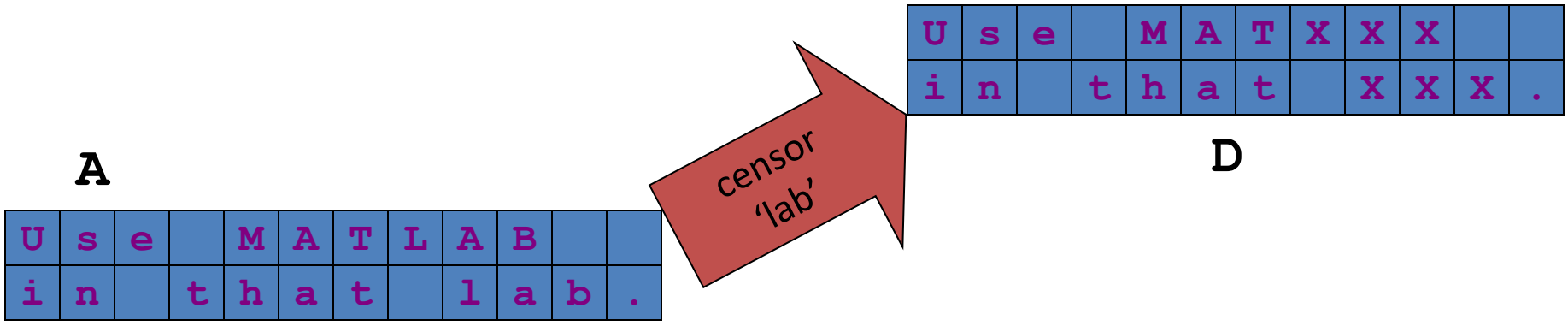
```
function up = toUpper(cha)
% up is the upper case of character cha.
% If cha is not a letter then up is just cha.

up= cha;

if ( cha >= 'a' && cha <= 'z' )

    % Find distance of cha from ‘a’




end
```

```
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;

if ( cha >= 'a' && cha <= 'z' )

    % Find distance of cha from ‘a’
    offset= cha - 'a';

    % Go same distance from ‘A’

end
```

```
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;

if ( cha >= 'a' && cha <= 'z' )

    % Find distance of cha from ‘a’
    offset= cha - 'a';

    % Go same distance from ‘A’
    up= char('A' + offset);
end
```

Example: censoring words

```
function D = 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.
% D is A with X's replacing str.
```



```
function D = 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.
% D is A with X's replacing the censored string str.

D= 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
```

```
function D = 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.
% D is A with X's replacing the censored string str.

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

% Build a string of X's of the right length
Xs= char( zeros(1,ns));
for k= 1:ns
    Xs(k)= 'X';
end

% Traverse the matrix to censor string str
```

zeros returns an array of type **double**

```
function D = 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.
% D is A with X's replacing the censored string str.

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

% Build a string of X's of the right length
Xs= char( zeros(1,ns));
for k= 1:ns
    Xs(k)= 'X';
end

% Traverse the matrix to censor string str
for r= 1:nr
    for c= 1:nc-ns+1
        if  strcmp( s , B(r, c:c+ns-1) )==1
            D(r, c:c+ns-1)= Xs;
        end
    end
end
```

```
function D = 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.
% D is A with X's replacing the censored string str.

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

% Build a string of X's of the right length
Xs= char( zeros(1,ns));
for k= 1:ns
    Xs(k)= 'X';
end

% Traverse the matrix to censor string str
for r= 1:nr
    for c= 1:nc-ns+1
        if  strcmp( s , B(r, c:c+ns-1) )==1
            D(r, c:c+ns-1)= Xs;
        end
    end
end
```

zeros returns an array of type **double**