

- Previous Lecture:

- Characters and strings

- Today's Lecture:

- Cell arrays

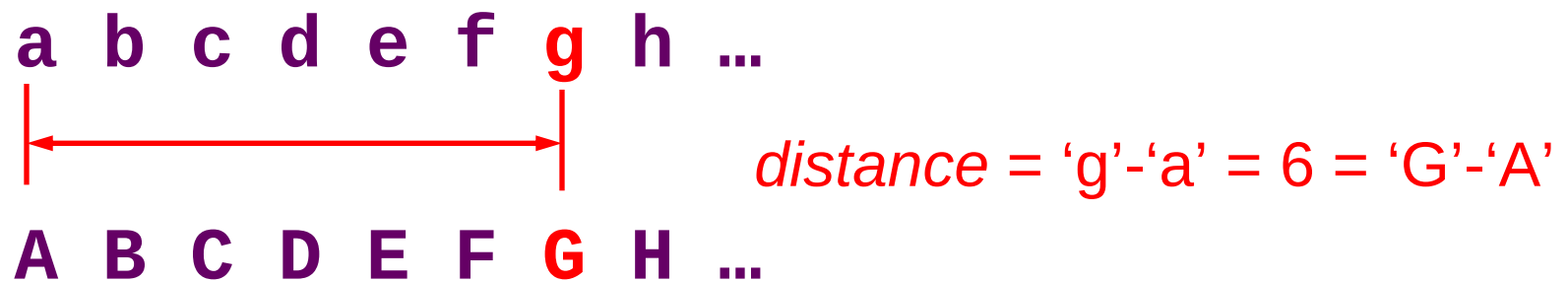
- Announcement:

- Project 5 due Thursday 10/5 at 11pm
- Prelim 2 on Nov 10. Email Randy Hess (rbhess@cs.cornell.edu) if you have a conflict.

Example: toUpper

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

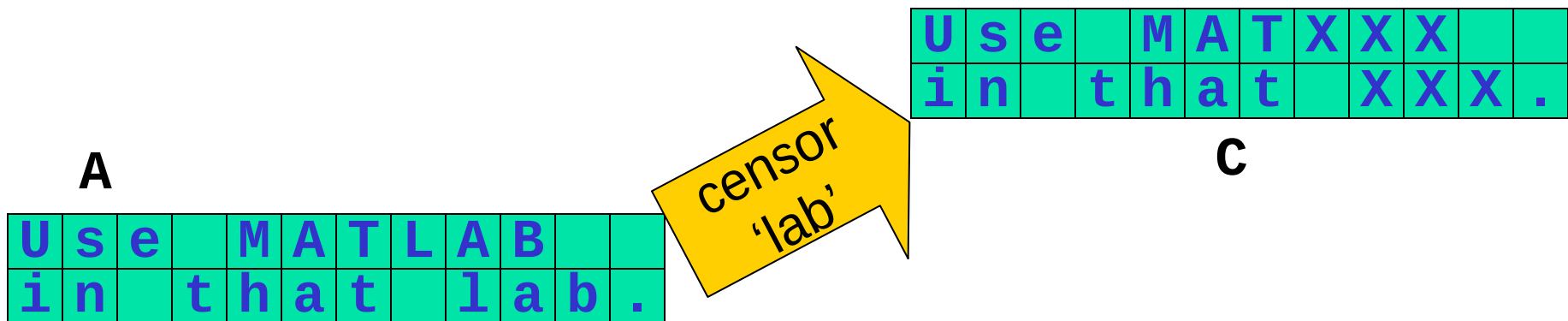
Hint: Think about the **distance** between a letter and the base letter 'a' (or 'A'). E.g.,



Of course, do not use Matlab function `upper`!

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.
```



Matrix vs. Cell Array

Vectors and matrices store values of the same type in all components

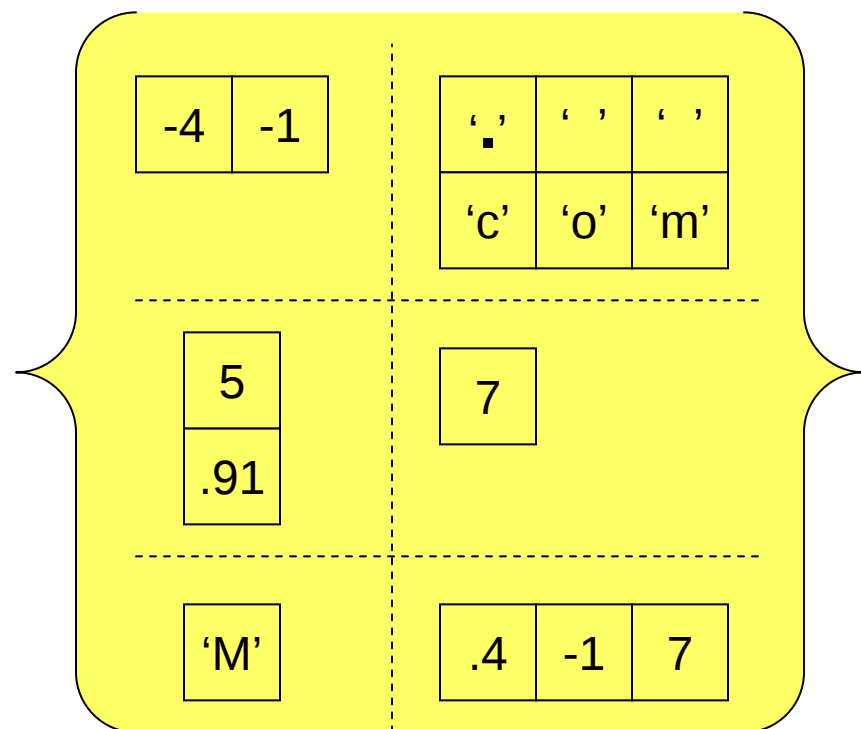
3.1
2
-1
9
1.1

5 x 1
matrix

'c'	'o'	'm'	' '	's'
'1'	'1'	'1'	'2'	' '
'M'	'a'	't'	' '	' '
' '	' '	'L'	'A'	'B'

4 x 5
matrix

A cell array is a special array whose individual components may contain different types of data



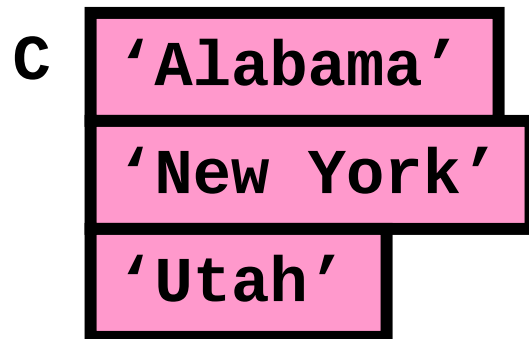
3 x 2 cell array

Cell Arrays of Strings

C = { 'Alabama', 'New York', 'Utah' }



C = { 'Alabama'; 'New York'; 'Utah' }



**M = ['Alabama'; ...
 'New York'; ...
 'Utah']**

M

'A'	'l'	'a'	'b'	'a'	'm'	'a'	' '
'N'	'e'	'w'	' '	'Y'	'o'	'r'	'k'
'U'	't'	'a'	'h'	' '	' '	' '	' '

Use braces { } for creating/addressing cell arrays

Matrix

- Create/append

```
m= [ 5, 4 ; ...  
      1, 2 ; ...  
      0, 8 ]
```

- Addressing

```
m(2,1)= pi
```

Cell Array

- Create/append

```
C= { ones(2,2), 4 ; ...  
      'abc' , ones(3,1) ; ...  
      9 , 'a cell' }
```

- Addressing

```
C{2,1}= 'ABC'  
C{3,2}= pi  
disp(C{3,2})
```

Creating cell arrays...

```
C= {'Oct', 30, ones(3,2)};
```

is the same as

```
C= cell(1,3); % not necessary
```

```
C{1}= 'Oct';
```

```
C{2}= 30;
```

```
C{3}= ones(3,2);
```

You can assign the empty cell array: **D = {}**

Example: Represent a deck of cards with a cell array

D{1} = 'A Hearts';

D{2} = '2 Hearts';

:

D{13} = 'K Hearts';

D{14} = 'A Clubs';

:

D{52} = 'K Diamonds';

But we don't want to have to type all combinations of suits and ranks in creating the deck... How to proceed?

Make use of a suit array and a rank array ...

```
suit = { 'Hearts', 'Clubs', ...  
        'Spades', 'Diamonds' };  
  
rank = { 'A', '2', '3', '4', '5', '6', ...  
        '7', '8', '9', '10', 'J', 'Q', 'K' };;
```

Then concatenate to get a card. E.g.,

```
str = [rank{3} ' ' suit{2} ];  
D{16} = str;
```

So D{16} stores '3 Clubs'

To get all combinations, use **nested loops**

```
i = 1;  % index of next card

for k= 1:4
    % Set up the cards in suit k
    for j= 1:13
        D{i} = [ rank{j} ' ' suit{k} ];
        i = i+1;
    end
end
```

See function **CardDeck**

Example: deal a 12-card deck

D:            

N:    **1, 5, 9** **$4k - 3$**

E:    **2, 6, 10** **$4k - 2$**

S:    **3, 7, 11** **$4k - 1$**

W:    **4, 8, 12**

$4k$

```
% Deal a 52-card deck
```

```
N = cell(1,13); E = cell(1,13);
```

```
S = cell(1,13); W = cell(1,13);
```

```
for k=1:13
```

```
    N{k} = D{4*k-3};
```

```
    E{k} = D{4*k-2};
```

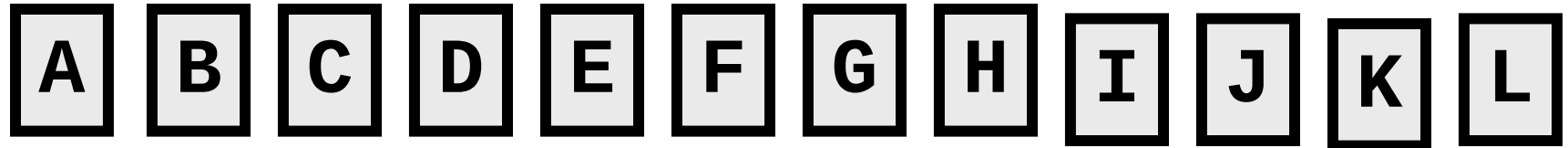
```
    S{k} = D{4*k-1};
```

```
    W{k} = D{4*k};
```

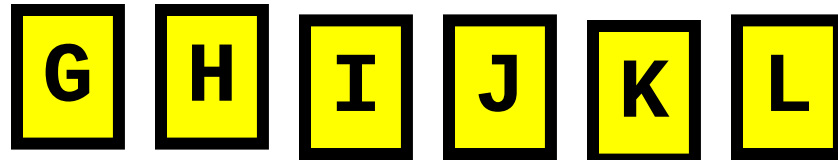
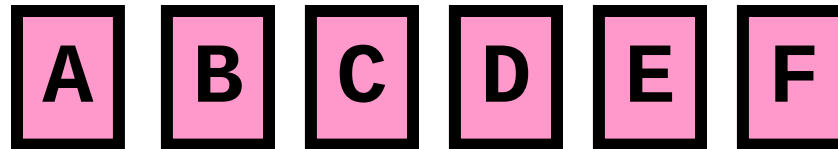
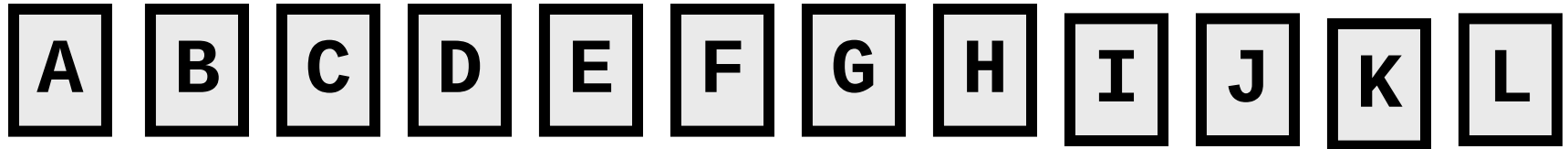
```
end
```

See function **Deal**

The “perfect shuffle” of a 12-card deck



Step 1: Cut the deck



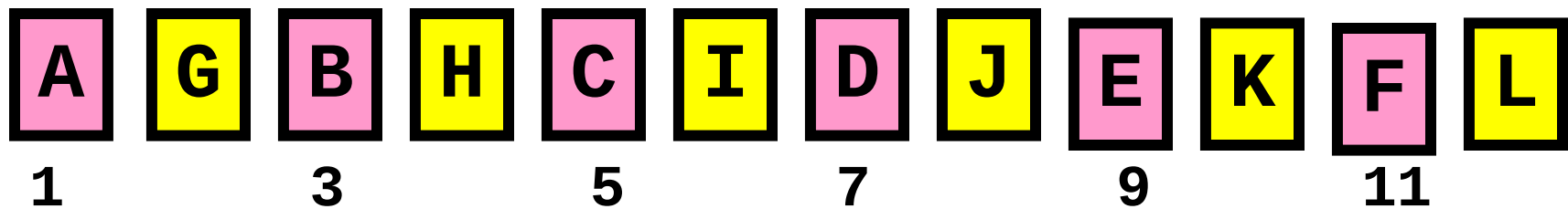
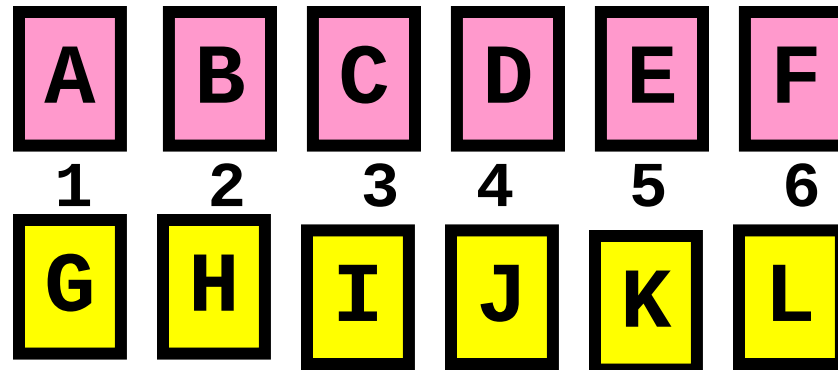
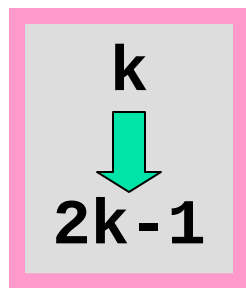
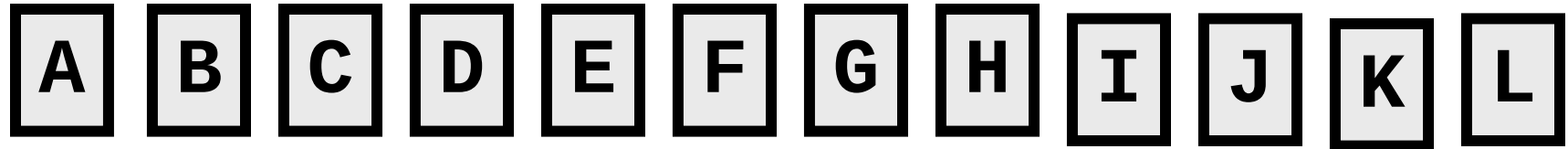
Step 2: Alternate

A B C D E F G H I J K L

A B C D E F
1 2 3 4 5 6
G H I J K L

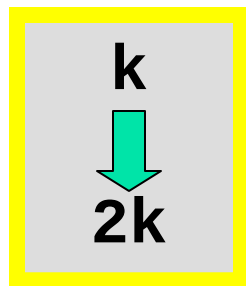
A G B H C I D J E K F L
1 2 3 4 5 6 7 8 9 10 11 12

Step 2: Alternate



Step 2: Alternate

A B C D E F G H I J K L

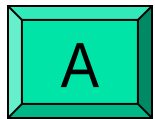


A B C D E F
1 2 3 4 5 6
G H I J K L

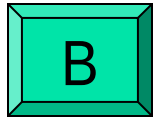
A G B H C I D J E K F L
2 4 6 8 10 12

Shuffle.m

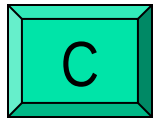
How many perfect shuffles must one make to get back to the original arrangement of the cards?



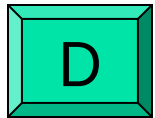
Cannot cycle back to original arrangement



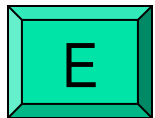
2



4



8



16

See script **ShowCards**

Example: subset of clicker IDs

IDs

```
['d091314'; ...  
'h134d83'; ...  
'h4567s2'; ...  
'fr83209']
```

Find subset that
begins with 'h'



L

```
{'h134d83'; ...  
'h4567s2'}
```

```
L= {};  
k= 0;  
for r=1:size(ID,1)  
    if IDs(r,1)=='h'  
        k= k+1;  
        L{k }= IDs(r,:);  
    end  
end
```

Directly assign into a
particular cell—good!

```
L= {};  
  
for r=1:size(ID,1)  
    if IDs(r,1)=='h'  
  
        L= [L; IDs(r,:)];  
    end  
end
```

Concatenate cells or
cell arrays—prone to
problems!

Example: Build a cell array of Roman numerals
for 1 to 3999

C{1} = 'I'

C{2} = 'II'

C{3} = 'III'

:

C{2007} = 'MMVII'

:

C{3999} = 'MMMCMXCIX'