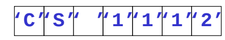
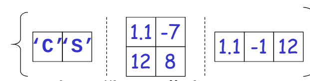
- Previous Lecture:
  - Characters and strings
- Today's Lecture:
  - Cell arrays
- Announcements:
  - Discussion this week will be in Upson B7 lab

Array vs. Cell Array

- Simple array



  - Each component stores one scalar. E.g., one char, one double, or one uint8 value
  - All components have the same type
- Cell array

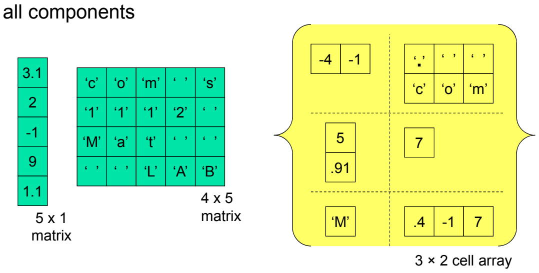


  - Each cell can store something "bigger" than one scalar, e.g., a vector, a matrix, a string (vector of chars)
  - The cells may store items of different types

1-d and 2-d examples …

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

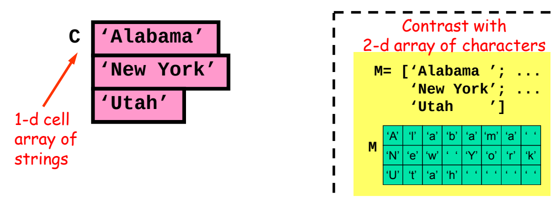
Cell array: individual components may contain different types of data



Cell Arrays of Strings

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


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



Use braces { } for creating and addressing cell arrays

Matrix	Cell Array
Create	Create
m= [5, 4 ; … 1, 2 ; … 0, 8]	C= { ones(2,2), 4 ; … 'abc' , ones(3,1) ; … 9 , 'a cell' }
Addressing	Addressing
m(2,1)= pi	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

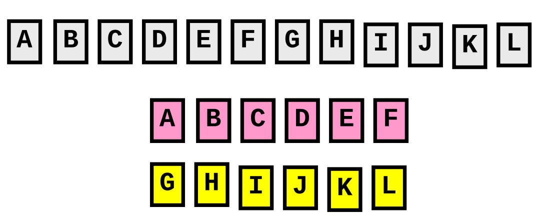
```
i = 1;  % index of next card

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**

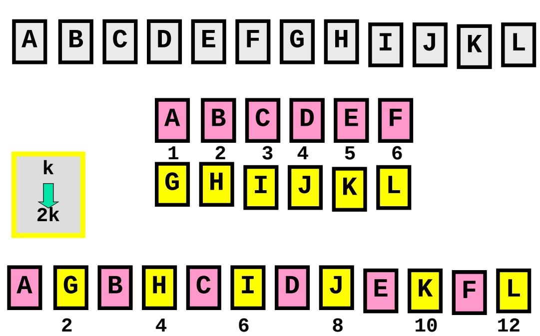

```
% Deal a 52-card deck
N = cell(1,13); E = cell(1,13);
S = cell(1,13); W = cell(1,13);

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**

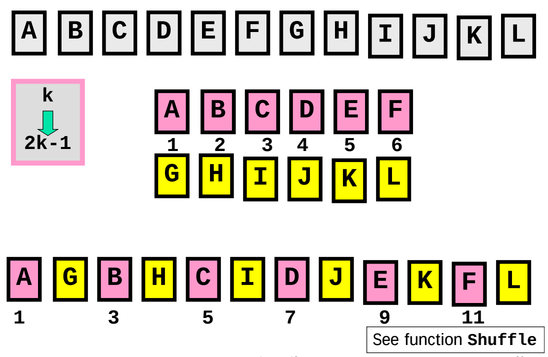
Perfect Shuffle, Step 1: cut the deck



Perfect Shuffle, Step 2: Alternate



Perfect Shuffle, Step 2: Alternate



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

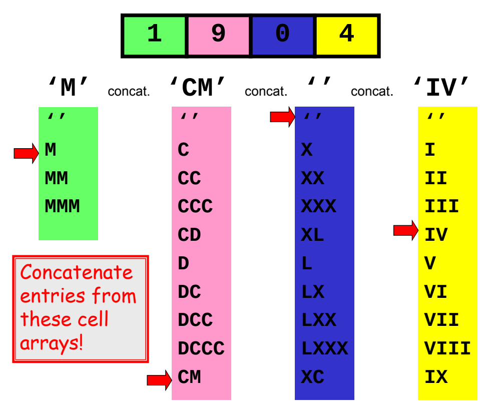
```
C{1} = 'I'
C{2} = 'II'
C{3} = 'III'
   :
C{2007} = 'MMVII'
   :
C{3999} = 'MMMXMXCIX'
```


Example

1904 = 1*1000 + 9*100 + 0*10 + 4*1

= M CM IV

= MCMIV



Ones-Place Conversion

```
function r = Ones2R(x)
% x is an integer that satisfies
%      0 <= x <= 9
% r is the Roman numeral with value x.

Ones = {'I', 'II', 'III', 'IV', ...
        'V', 'VI','VII', 'VIII', 'IX'};

if x==0
     r = '';
else
     r = Ones{x};
end
```


Similarly, we can implement these functions:

```
function r = Tens2R(x)
% x is an integer that satisfies
%      0 <= x <= 9
% r is the Roman numeral with value 10*x.
```

```
function r = Hund2R(x)
% x is an integer that satisfies
%      0 <= x <= 9
% r is the Roman numeral with value 100*x
```

```
function r = Thou2R(x)
% x is an integer that satisfies
%      0 <= x <=3
% r is the Roman numeral with value 1000*x
```

Now we can build the Roman numeral cell array for 1,…,3999

```
for a = 0:3  % possible values in thous place
  for b = 0:9  % values in hundreds place
    for c = 0:9  % values in tens place
      for d = 0:9  % values in ones place
        n = a*1000 + b*100 + c*10 + d;
        if n>0
          C{n} = [Thou2R(a) Hund2R(b)...
                  Tens2R(c) Ones2R(d)];
        end
      end
    end
  end
end
```

Four strings concatenated together

The n^{th} component of cell array C

Lecture 18 39

The reverse conversion problem

Given a Roman Numeral, compute its value.
Assume cell array C(3999,1) available where:

```
C{1} = 'I'
C{2} = 'II'
  :
C{3999} = 'MMMCMXCIX'
```

See script **RN2Int**

Lecture 18 40

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(IDs,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!