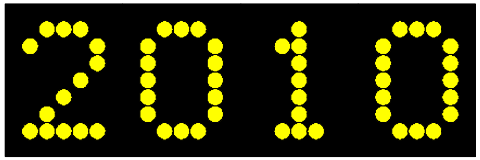
- Previous Lecture:
  - Structure, structure array
  - Working with data files

- Today's Lecture:
  - Another data file example; built-in `sort` function
  - Review matrix, cell array, structure array



- Announcement:
  - Discussion in classrooms this week
  - Prelim 3 on Tues, Apr 19th, at 7:30pm
  - Project 5 due Thurs 11pm?? Will depend on class vote.
  - Project 5
    - readZipcodes returns two struct arrays, i.e., *each component in the array is a struct.*
    - How to print a '%'? `fprintf('One percent sign: %% \n')`

A detailed sort-a-file example

Suppose each line in the file

`statePop.txt`

is structured as follows:

Cols 1-14: State name

Cols 16-24: Population (millions)

The states appear in alphabetical order.

Alabama	4557808
Alaska	663661
Arizona	5939292
Arkansas	2779154
California	36132147
Colorado	4665177
:	:
:	:
Texas	22859968
Utah	2469585
Vermont	623050
Virginia	7567465
Washington	6287759
West Virginia	1816856
Wisconsin	5536201
Wyoming	509294

A detailed sort-a-file example

Create a new file

`statePopSm2Lg.txt`

that is structured the same as **`statePop.txt`**

except that *the states are ordered from smallest to largest according to population.*

```
Alabama          4557808
Alaska            663661
Arizona          5939292
Arkansas         2779154
California      36132147
Colorado         4665177
   :                :
   :                :
```

- Need the pop as *numbers* for sorting.
- Can't just sort the pop—have to maintain association with the state names.

First, get the populations into an array

```
C = file2cellArray('StatePop');
n = length(C);
pop = zeros(n,1);
for i=1:n
  S = C{i};
  pop(i) = str2double(S(16:24));
end
```

```
function CA = file2cellArray(fname)
% fname is a string that names a .txt file
%   in the current directory.
% CA is a cell array with CA{k} being the
%   k-th line in the file.

fid= fopen([fname '.txt'], 'r');
k= 0;
while ~feof(fid)
   k= k+1;
   CA{k}= fgetl(fid);
end
fclose(fid);
```

First, get the populations into an array

```
C = file2cellArray('StatePop');
n = length(C);
pop = zeros(n,1);
for i=1:n
   S = C{i};
   pop(i) = str2double(S(16:24));
end
```

C

{'Alab 4558000'
'Alas 664000'
⋮
'Cali 36132000'

'Verm 623000'
⋮
'Wyom 509000'}

cell array
of strings
in alpha. order

Cnew

{'Wyom 509000'
'Verm 623000'
.
.
.
'Cali 36132000'}

C

{ 'Alab 4558000'
'Alas 664000'
⋮
'Cali 36132000'
'Verm 623000'
⋮
'Wyom 509000' }

cell array
of strings
in alpha. order

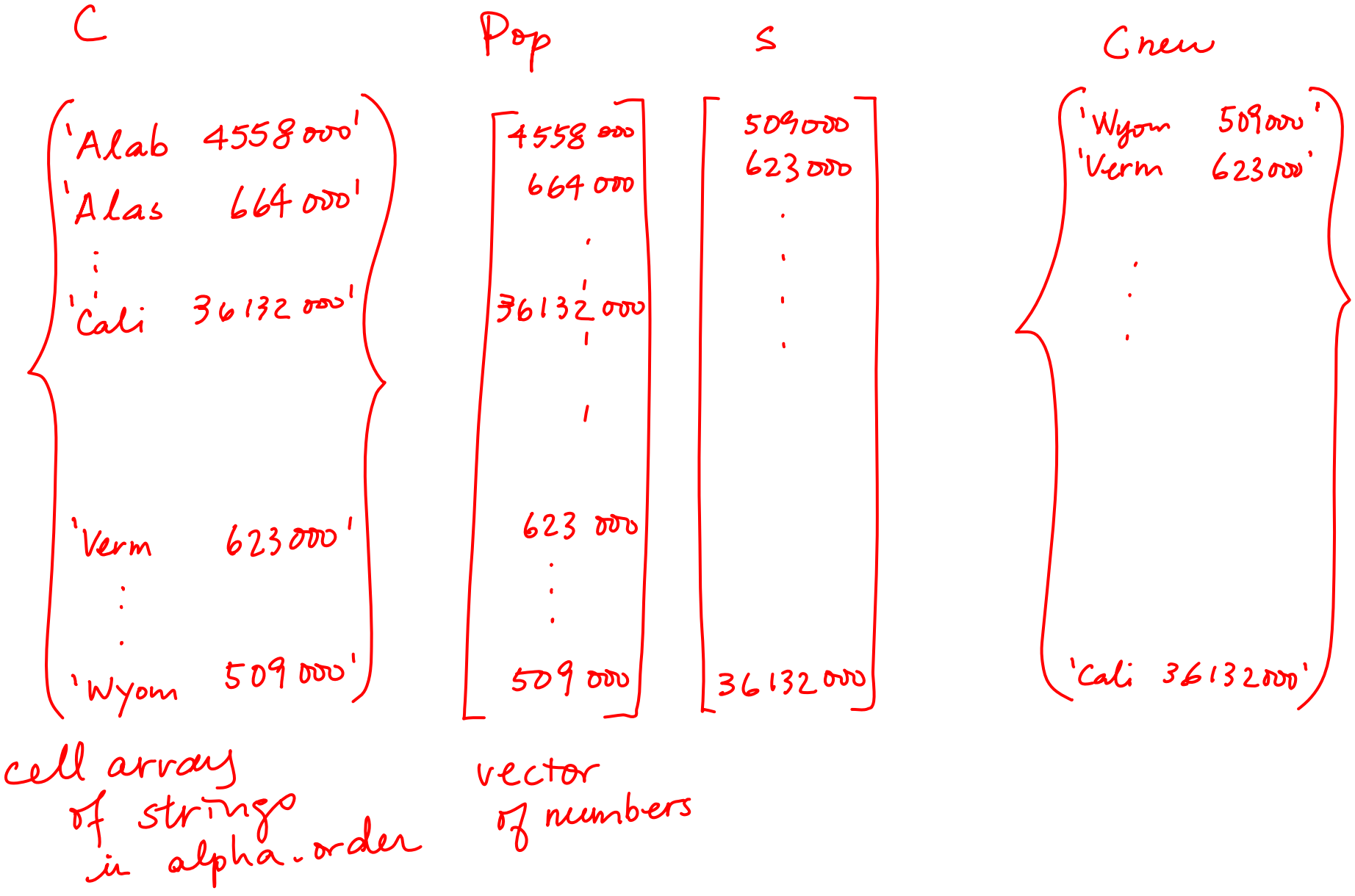
Pop

[4558000
664000
⋮
36132000
⋮
623000
⋮
509000]

vector
of numbers

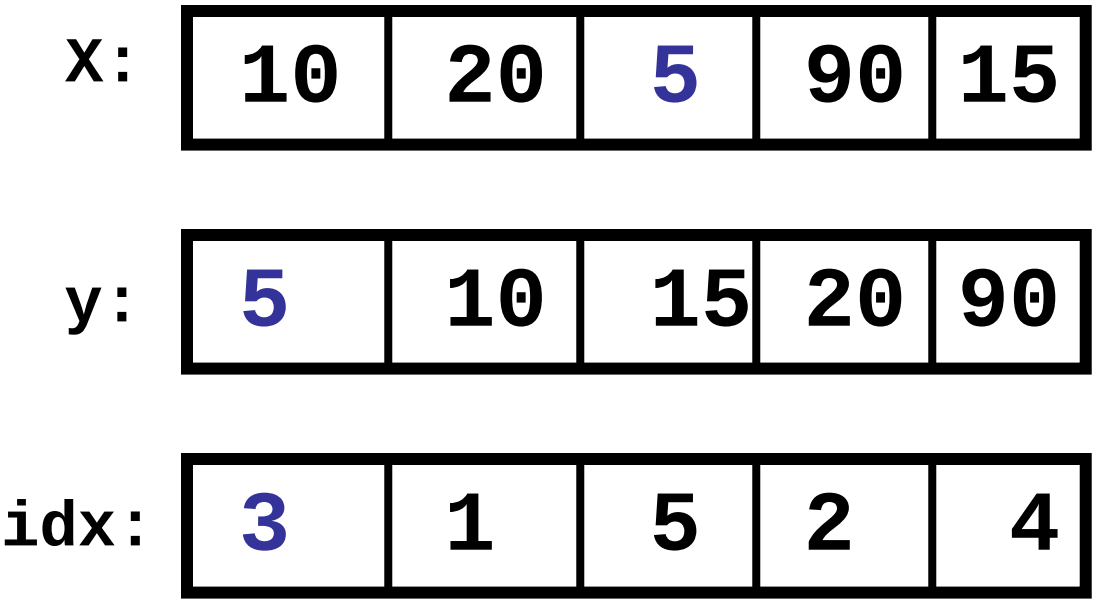
Cnew

{ 'Wyom 509000'
'Verm 623000'
⋮
'Cali 36132000' }


C
'Alab 4558000'
'Alas 664000'
'Cali 36132000'
'Verm 623000'
'Wyom 509000'
Pop
4558000
664000
36132000
623000
509000
S
5090000
623000
36132000
Cnew
'Wyom 509000'
'Verm 623000'
'Cali 36132000'
cell array of strings in alpha. order
vector of numbers

Built-In function `sort`

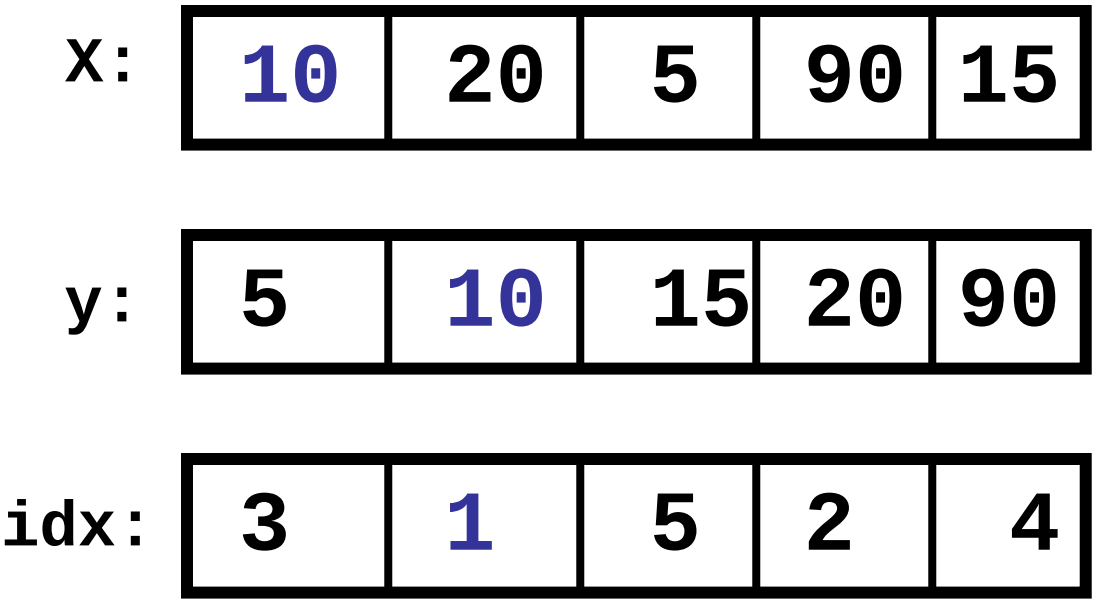
Syntax: `[y,idx] = sort(x)`



`y(1) = x(3) = x(idx(1))`

Built-In function `sort`

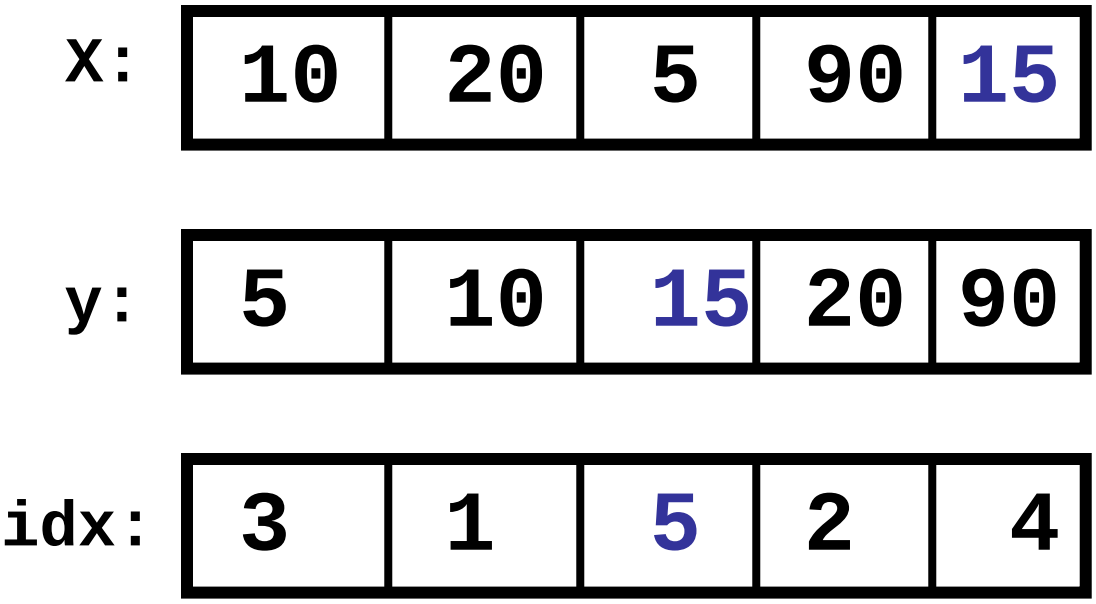
Syntax: `[y,idx] = sort(x)`




```
y(2) = x(1) = x(idx(2))
```

Built-In function `sort`

Syntax: `[y,idx] = sort(x)`



`y(3) = x(5) = x(idx(3))`

Built-In function `sort`

Syntax: `[y,idx] = sort(x)`

x:	10	20	5	90	15

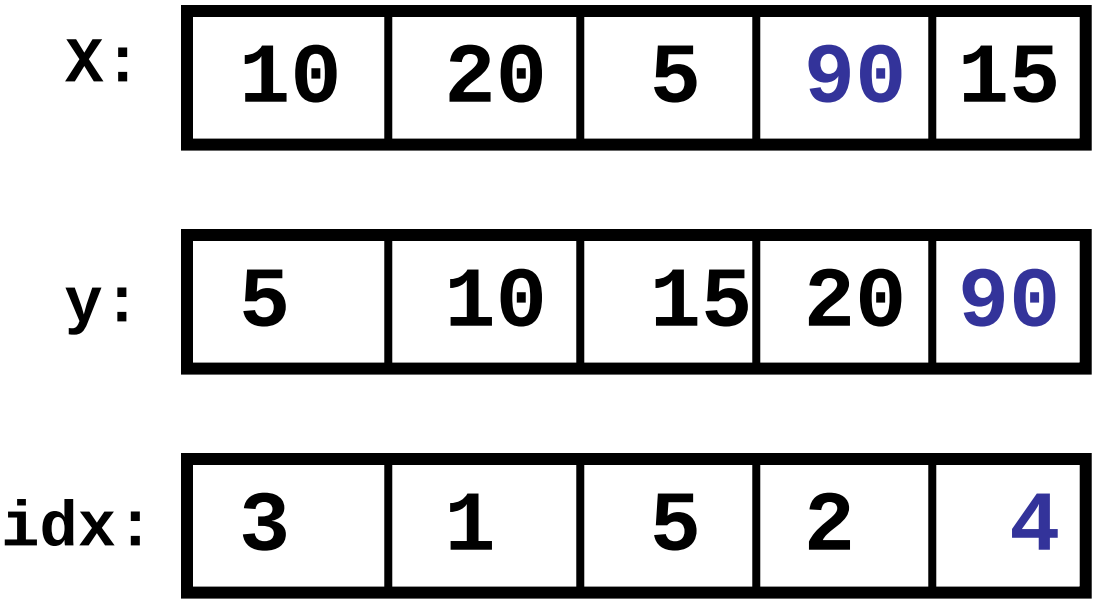
y:	5	10	15	20	90

idx:	3	1	5	2	4

`y(4) = x(2) = x(idx(4))`

Built-In function `sort`

Syntax: `[y,idx] = sort(x)`



`y(5) = x(4) = x(idx(5))`

Built-In function `sort`

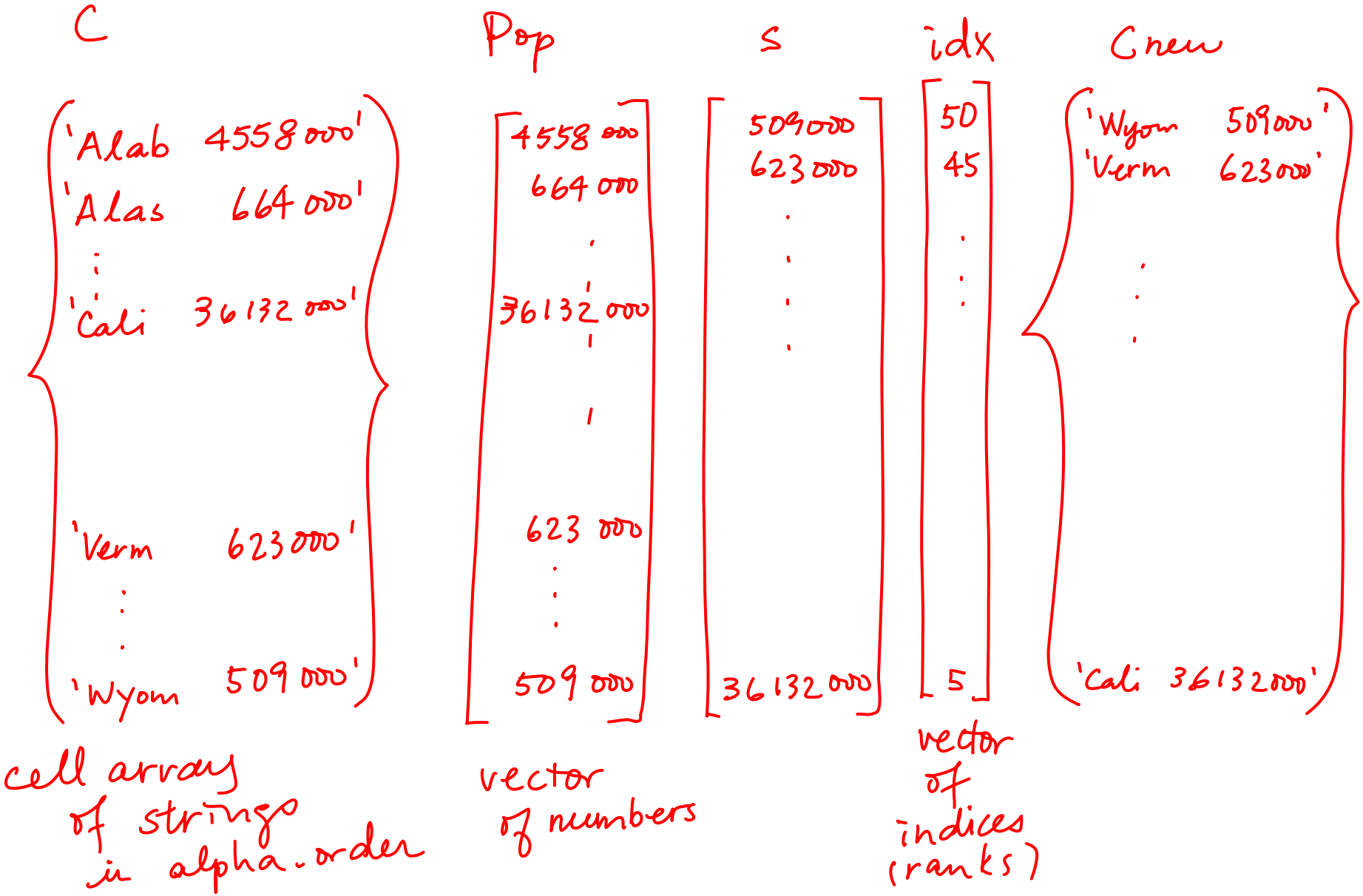
Syntax: `[y,idx] = sort(x)`

x:	10	20	5	90	15

y:	5	10	15	20	90

idx:	3	1	5	2	4

```
y(k) = x(idx(k))
```

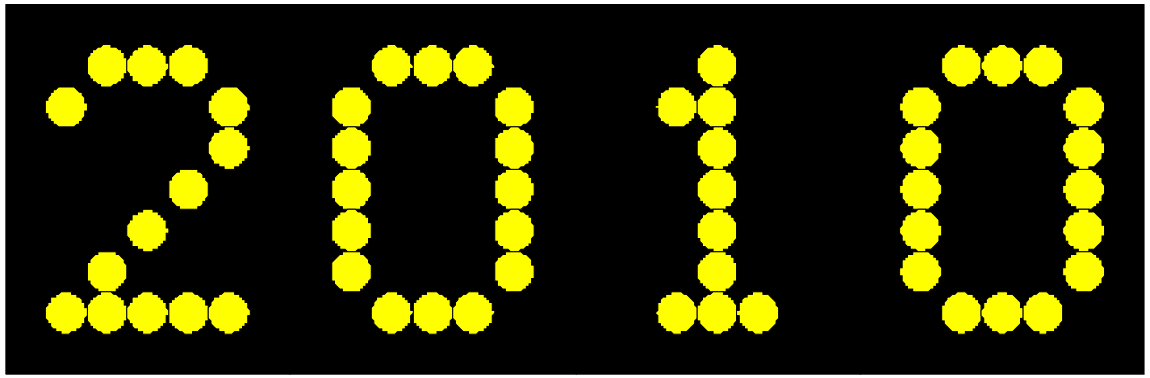

C
Pop
s
idx
Cnew
'Alab 4558000'
'Alas 664000'
'Cali 36132000'
'Verm 623000'
'Wyom 509000'
4558000
664000
36132000
623000
509000
509000
623000
36132000
50
45
5
'Wyom 509000'
'Verm 623000'
'Cali 36132000'
cell array of strings in alpha. order
vector of numbers
vector of indices (ranks)

Sort from little to big

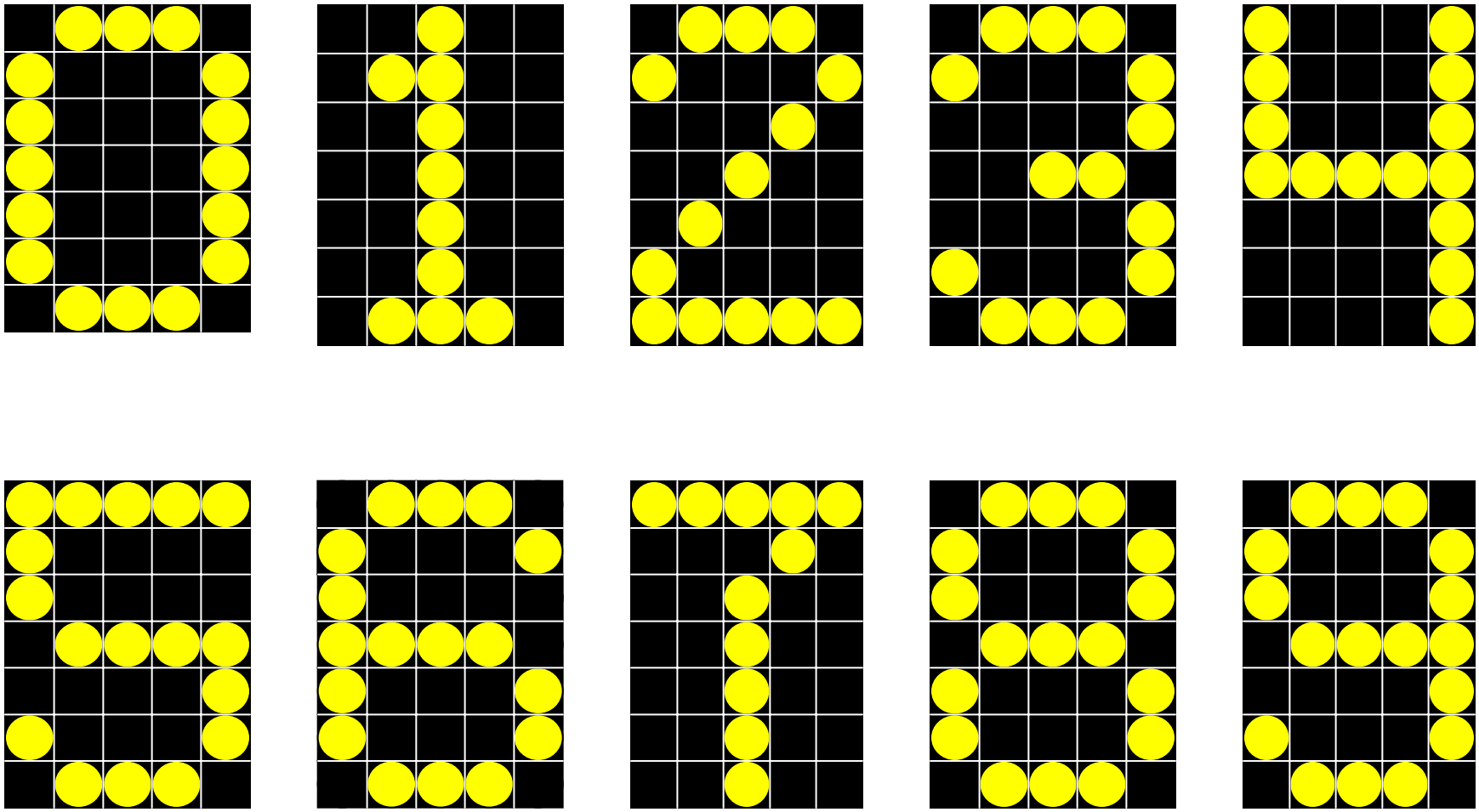
```
% C is cell array read from statePop.txt
% pop is vector of state pop (numbers)
[s,idx] = sort(pop);
Cnew = cell(n,1);
for i=1:length(C)
    ithSmallest = idx(i);
    Cnew{i} = C{ithSmallest};
end

cellArray2file(Cnew,'statePopSm2Lg')
```

Wyoming	509294
Vermont	623050
North Dakota	636677
Alaska	663661
South Dakota	775933
Delaware	843524
Montana	935670
:	:
:	:
Illinois	12763371
Florida	17789864
New York	19254630
Texas	22859968
California	36132147

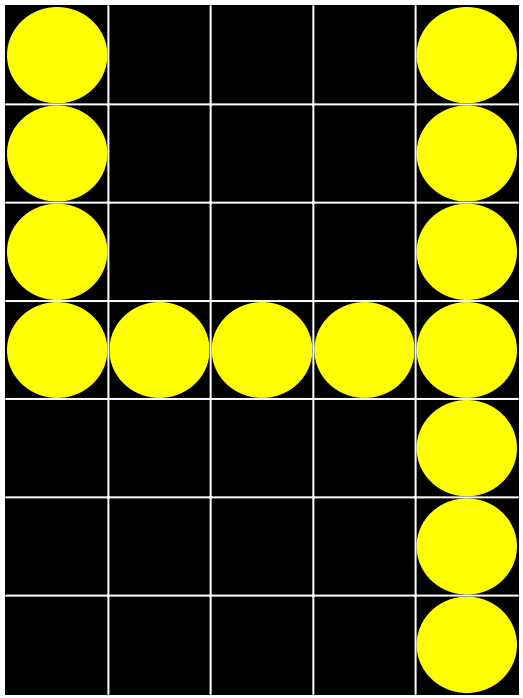
Application: digital displays



7-by-5 “dot matrices”



A “bit map” for each digit



A “light” is either on or off.

A 7-by-5 matrix of zeros and ones can “tell the whole story.”

Computing with these bitmaps

- What is a good scheme for storing the 10 bitmaps?
- How to draw one digit?
- How to display a number?
- Other interesting questions:
  - How to draw a mirror image of a digit?
  - Which "light bulb" switches on most often?

Design decisions…

How do we package a particular digit?

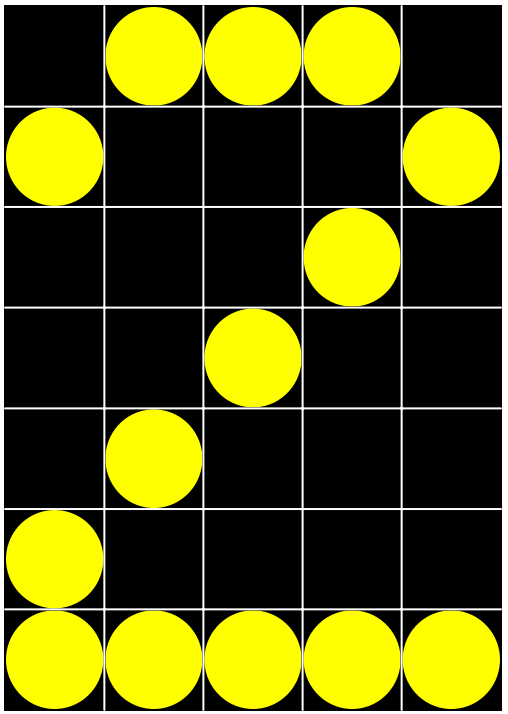
numerical array or character array

How do we package the collection of digits?

cell array or structure array

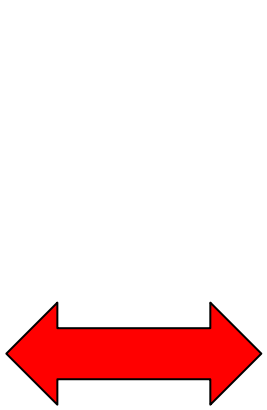
Can use a numerical array for each digit

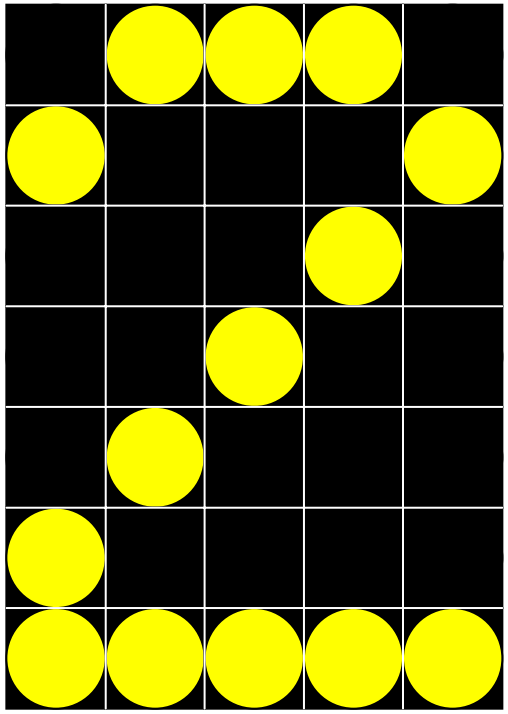
```
[ 0 1 1 1 0;…
  1 0 0 0 1;…
  0 0 0 1 0;…
  0 0 1 0 0;…
  0 1 0 0 0;…
  1 0 0 0 0;…
  1 1 1 1 1];
```



Can use a character array for each digit

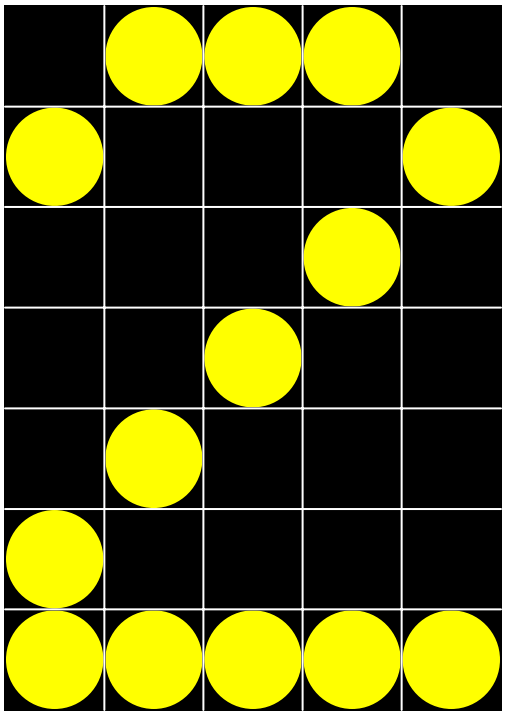
```
[ '01110';…
  '10001';…
  '00010';…
  '00100';…
  '01000';…
  '10000';…
  '11111'];
```





Can use a cell array to keep the 10 bitmaps

```
M = [ 0 1 1 1 0;…
      1 0 0 0 1;…
      0 0 0 1 0;…
      0 0 1 0 0;…
      0 1 0 0 0;…
      1 0 0 0 0;…
      1 1 1 1 1];
```



```
D{2} = M;
```

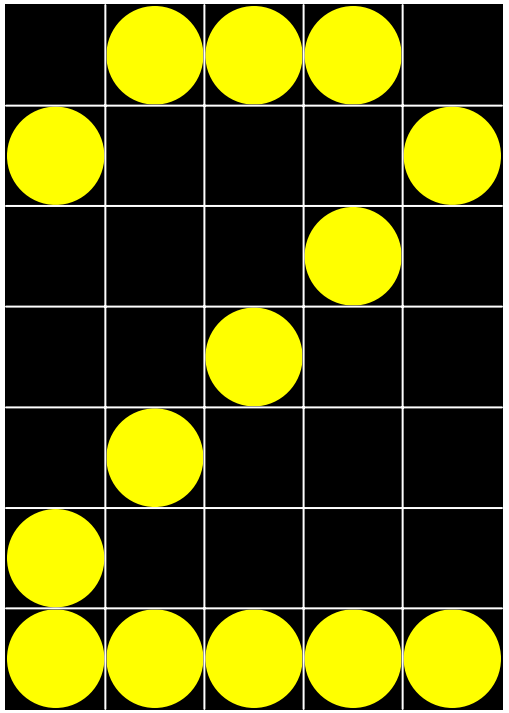
Each cell of cell array **D** is a numerical matrix.

With D{1},…,D{10} set up as a cell array of numerical matrices, can do computation as follows:

```
% given 1<=k<=10
M = D{k};
if M(4,3)==1
    disp('Middle light is on')
end
```

Still using a cell array to keep the 10 bitmaps

```
M = ['01110';…
     '10001';…
     '00010';…
     '00100';…
     '01000';…
     '10000';…
     '11111'];
```



```
D{2} = M;
```

Each cell of cell array **D** is a matrix of characters.

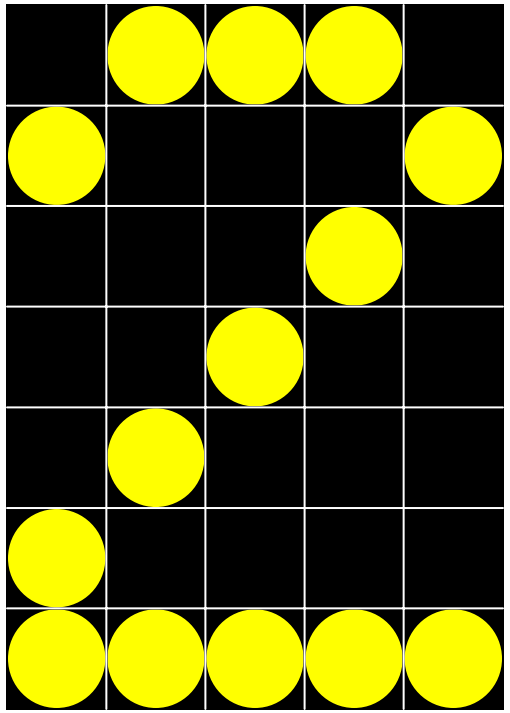
With D{1},…,D{10} set up as a cell array of character matrices, can do computation as follows:

```
% given 1<=k<=10
M = D{k};
if strcmp(M(4,3),'1')
    disp('Middle light is on')
end
```

```
['01110';…
 '10001';…
 '00010';…
 '00100';…
 '01000';…
 '10000';…
 '11111'];
```

Can use a structure array to keep the 10 bitmaps

```
M = [ 0 1 1 1 0;…
      1 0 0 0 1;…
      0 0 0 1 0;…
      0 0 1 0 0;…
      0 1 0 0 0;…
      1 0 0 0 0;…
      1 1 1 1 1];
```



```
D(2) = struct('map', M);
```

Each component of array **D** is a structure, and the sole field in the structure is a matrix of numbers.

Using a structure array to keep the 10 bitmaps…
The k-th component of array D encodes digit k.

```
% Example for the digit 2
A = [ 0 1 1 1 0;…
      1 0 0 0 1;…
      0 0 0 1 0;…
      0 0 1 0 0;…
      0 1 0 0 0;…
      1 0 0 0 0;…
      1 1 1 1 1];
D(2)= struct('map', A);
```

Which fragment on the right is correct given **1<=k<=10**?

A

```
M = D(k);
if M(4,3)==1
    disp('Middle light on')
end
```

B

```
M = D(k).map;
if M(4,3)==1
    disp('Middle light on')
end
```

C

```
M = D{k};
if M(4,3)==1
    disp('Middle light on')
end
```

D

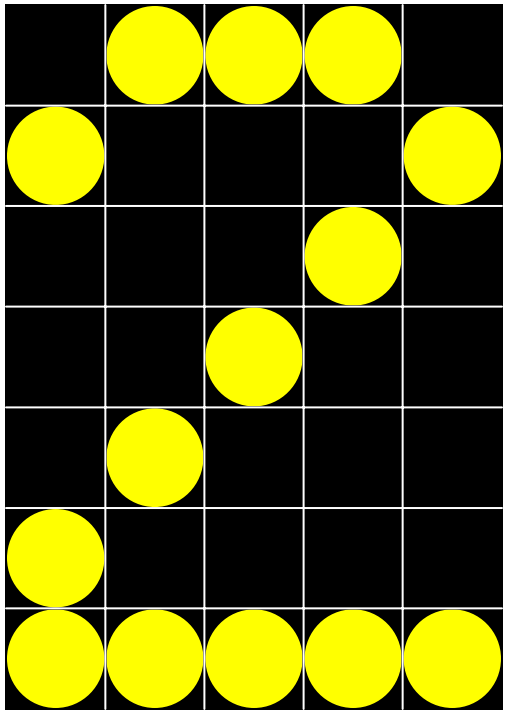
```
M = D{k}.map;
if M(4,3)==1
    disp('Middle light on')
end
```

With D(1),…,D(10) set up as a structure array of numerical matrices, can do computation as follows:

```
% given 1<=k<=10
M = D(k).map;
if M(4,3)==1
    disp('Middle light is on')
end
```

Can use a structure array to keep the 10 bitmaps

```
M = [ '01110';…
      '10001';…
      '00010';…
      '00100';…
      '01000';…
      '10000';…
      '11111'];
```



```
D(2) = struct('map', M);
```

Each component of array **D** is a structure, and the sole field in the structure is a matrix of characters.

With D(1),…,D(10) set up as a structure array of character matrices, can do computation as follows:

```
['01110';…
 '10001';…
 '00010';…
 '00100';…
 '01000';…
 '10000';…
 '11111'];
```

```
% given 1<=k<=10
M = D(k).map;
if strcmp(M(4,3),'1')
    disp('Middle light is on')
end
```

Choice for storing the bit maps

Cell array better than struct array

No point in having a structure with one field

Numerical array better than char array

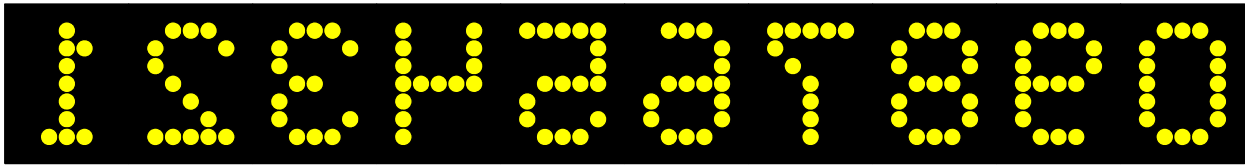
Plan on doing numerical computations with the bit maps—char arrays not handy

```
function D = TheDigits()
% D is a 10-by-1 cell array where D{k} is a 7-by-5 matrix
% that encodes digit k.  D{10} encodes 0.

D = cell(10,1);

D{1}  = [0 0 1 0 0;...
         0 1 1 0 0;...
         0 0 1 0 0;...
         0 0 1 0 0;...
         0 0 1 0 0;...
         0 0 1 0 0;...
         0 1 1 1 0];

D{2} =  [0 1 1 1 0;...
         1 0 0 0 1;...
         0 0 0 0 1;...
         0 0 0 1 0;...
         0 0 1 0 0;...
         0 1 0 0 0;...
         1 1 1 1 1];
⋮
```

Given this function, can write other functions to draw a single digit, multiple digits, etc.

See also **showDigits.m**

Produce a cell array of “reverse” digits



For every digit (matrix), need to reverse the order of the columns.

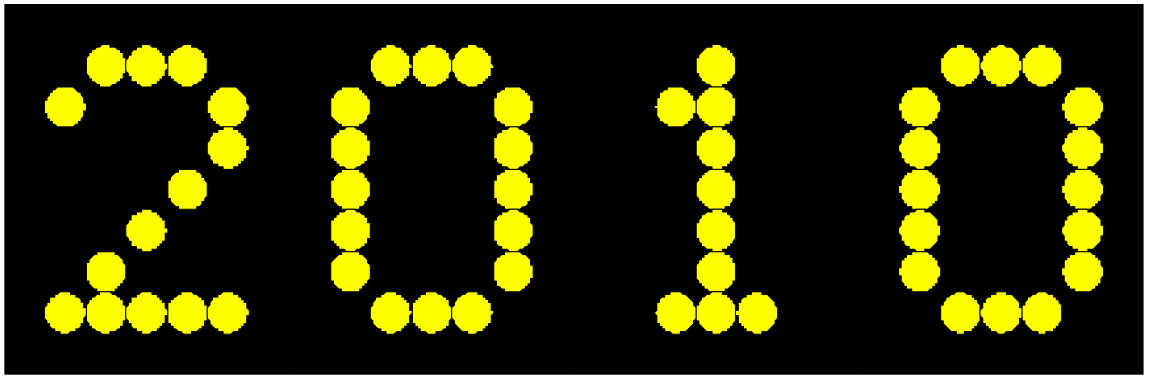
```
function B = reverseCol(A)
% B is a matrix obtained by reversing
% the order of the columns in matrix A

[nr, nc]= size(A);
B= zeros(nr,nc);
for k= 1:nc
    B(:,k) = A(:,nc-k+1);
end
```

```
function revD = reverseDigits()
% revD is a 10-by-1 cell array.
% revD{k} is the reversed 7-by-5 bitmap
% of digit k. revD{10} encodes 0.

D= TheDigits();
revD= cell(10,1);
```

```
function revD = reverseDigits()
% revD is a 10-by-1 cell array.
% revD{k} is the reversed 7-by-5 bitmap
% of digit k. revD{10} encodes 0.

D= TheDigits();
revD= cell(10,1);

for k= 1:length(D)
    M= D{k};
    revM= reverseCol(M);
    revD{k}= revM;
end
```

Digital display of a whole number

- Example: `showNumber(2010)`



- Need to convert the number to a vector of digits
  - `2010` → `[2 0 1 0]`
- Then display the digits in the vector side-by-side

```
function showNumber(n)
% Digital display of integer n, n>0

hold on; axis equal off

% Convert n to a vector of digits




% Display the digits in v
D = TheDigits();  % D{k} is matrix encoding digit k
```

```
function showNumber(n)
% Digital display of integer n, n>0

hold on; axis equal off

% Convert n to a vector of digits
```

```
% Display the digits in v
D = TheDigits();  % D{k} is matrix encoding digit k
for k=1:length(v)
    index= v(k);
    if index==0
        index= 10;
    end
    drawDigit(k,1,1,D{index})
end
```

```
function showNumber(n)
% Digital display of integer n, n>0

hold on; axis equal off

% Convert n to a vector of digits
v= [];
while n>0
    v= [rem(n,10) v];
    n= floor(n/10);
end


% Display the digits in v
D = TheDigits();  % D{k} is matrix encoding digit k
for k=1:length(v)
    index= v(k);
    if index==0
        index= 10;
    end
    drawDigit(k,1,1,D{index})
end
```

Discussion exercise

If you have an extra-long-life light bulb for your 7-by-5 display board, at which position would you install this light bulb?