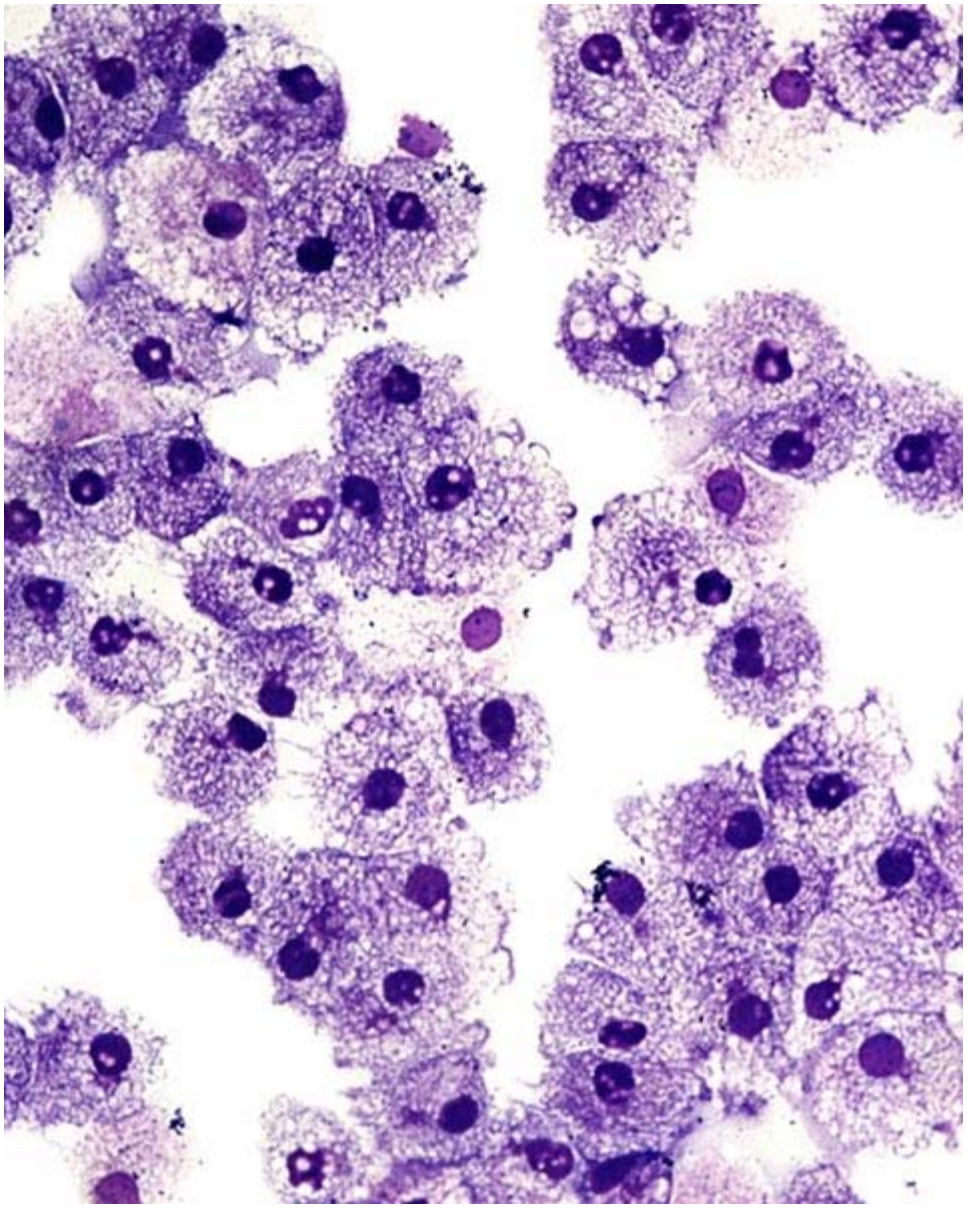


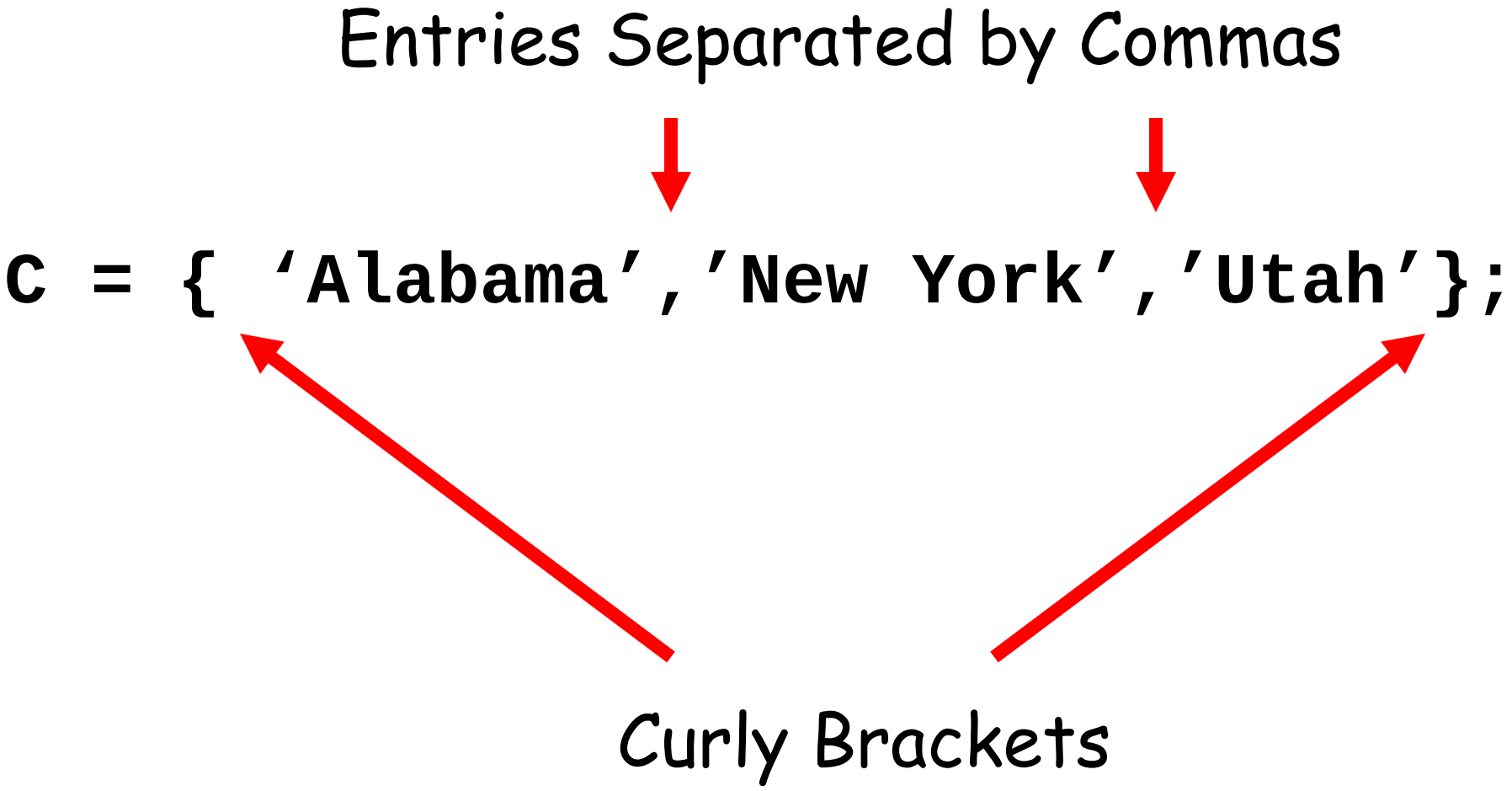
Cell Arrays

Lecture 18 (Mar 27)
CS100M – Spring 2008

A Small Cell Array…

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

Syntax


Entries Separated by Commas
C = { 'Alabama','New York','Utah'};
Curly Brackets

Another Way to Make a Cell Array

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

```
C = cell(1,3);
C{1} = 'Alabama';
C{2} = 'New York';
C{3} = 'Utah';
```

Application: Storing strings

Creating Vertical Cell Arrays

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

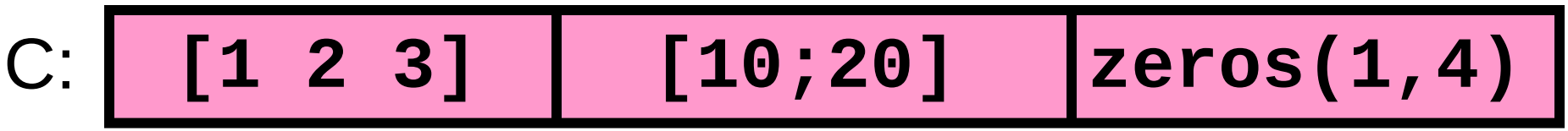
Semicolons

```
C = cell(3,1);
C{1} = 'Alabama';
C{2} = 'New York';
C{3} = 'Utah';
```

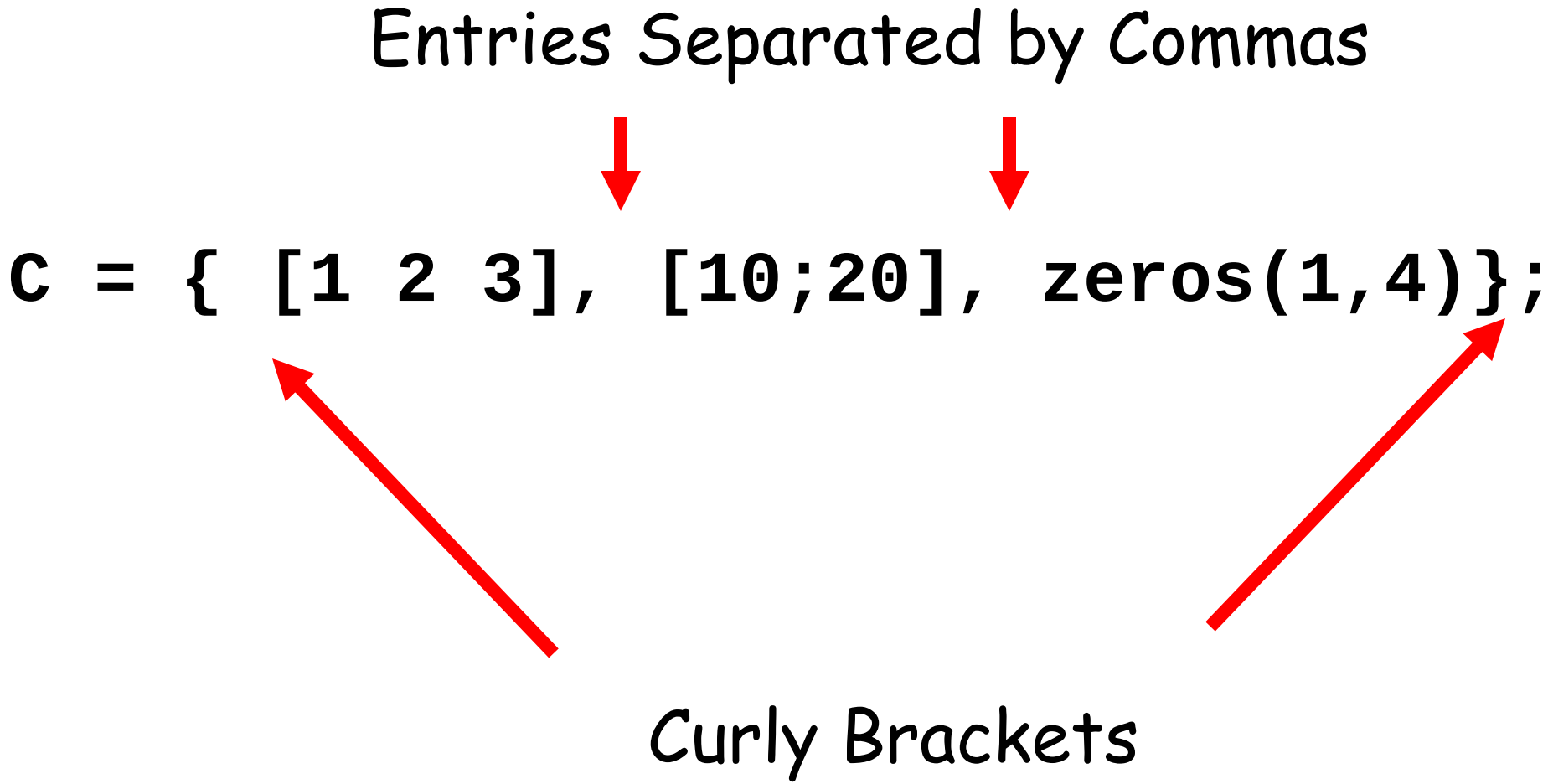
Three Rows, One Column

Another Small Cell Array…

```
C = { [1 2 3],[10;20], zeros(1,4)};
```



Syntax


Entries Separated by Commas
C = { [1 2 3], [10;20], zeros(1,4)};
Curly Brackets

Synonym

```
C = { [1 2 3], [10;20], zeros(1,4)};
```

```
C = cell(1,3);
C{1} = [1 2 3];
C{2} = [10;20];
C{3} = zeros(1,4);
```

Problem:
Set Up a Card Deck

Idea…

```
A{1} = ‘A Hearts’;
A{2} = ‘2 Hearts’;
          :
A{13} = ‘K Hearts’;
A{14} = ‘A Clubs’;
          :
A{52} = ‘K Diamonds’;
```

Initializations…

```
suit = {‘Hearts’, ‘Clubs’, …
        ‘Spades’, ‘Diamonds’};

rank = {‘A’,‘2’,’3’,’4’,’5’,’6’,…
    ’7’,’8’,’9’,’10’,’J’,’Q’,’K’};

A = cell(1,52);
```

Use Concatenation…

```
suit = {‘Hearts’, ‘Clubs’, …
        ‘Spades’, ‘Diamonds’};

rank = {‘A’,‘2’,’3’,’4’,’5’,’6’,…
    ’7’,’8’,’9’,’10’,’J’,’Q’,’K’};

A{16} = [rank{3} ‘ ‘ suit{2} ]
```

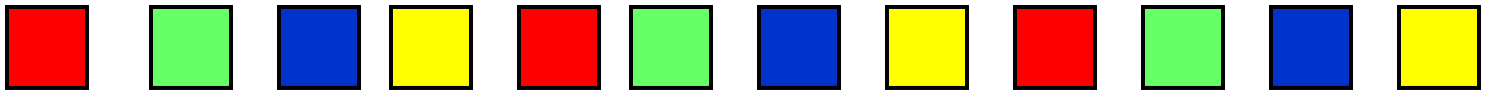
A{16} = ‘3 Clubs’

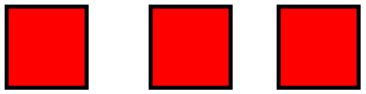
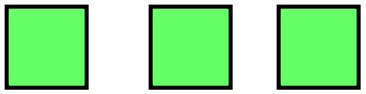
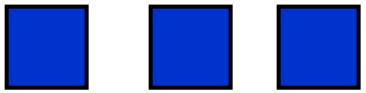
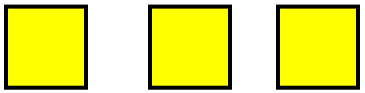
Nested Loops to Get All Combinations…

```
% i is index of next card…
  i = 1;
  for k=1:4
% Set up the cards in suit k
     for j=1:13
        A{i} = [ rank{j} ' ' suit{k} ];
        i = i+1
     end
  end
```

Problem:
Deal a Card Deck

Deal a Length-12 Card Deck

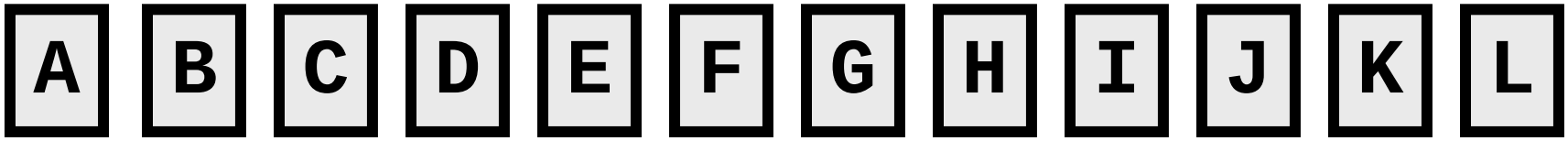
A: 

N: 	1,5,9	4k-3
E: 	2,6,10	4k-2
S: 	3,7,11	4k-1
W: 	4,8,12	4k

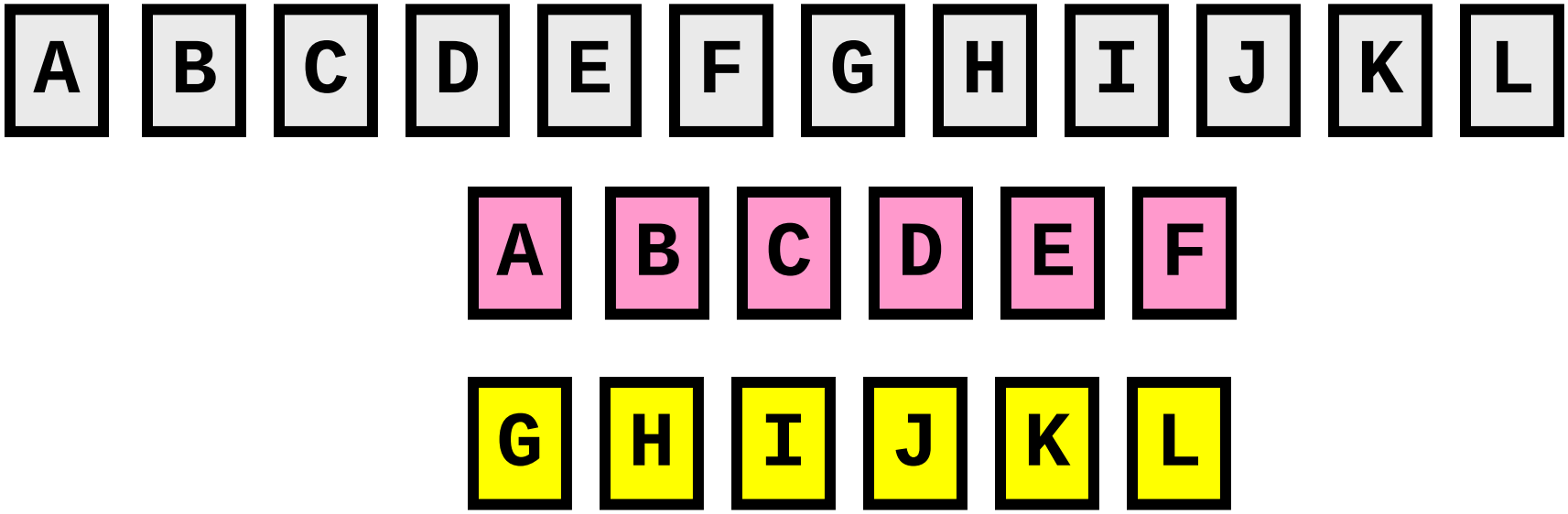
```
N = cell(1,13); E = cell(1,13);
S = cell(1,13); W = cell(1,13);

 for k=1:13
    N{k} = A{4*k-3};
    E{k} = A{4*k-2};
    S{k} = A{4*k-1};
    W{k} = A{4*k};
 end
```

Problem:
Shuffle a Card Deck

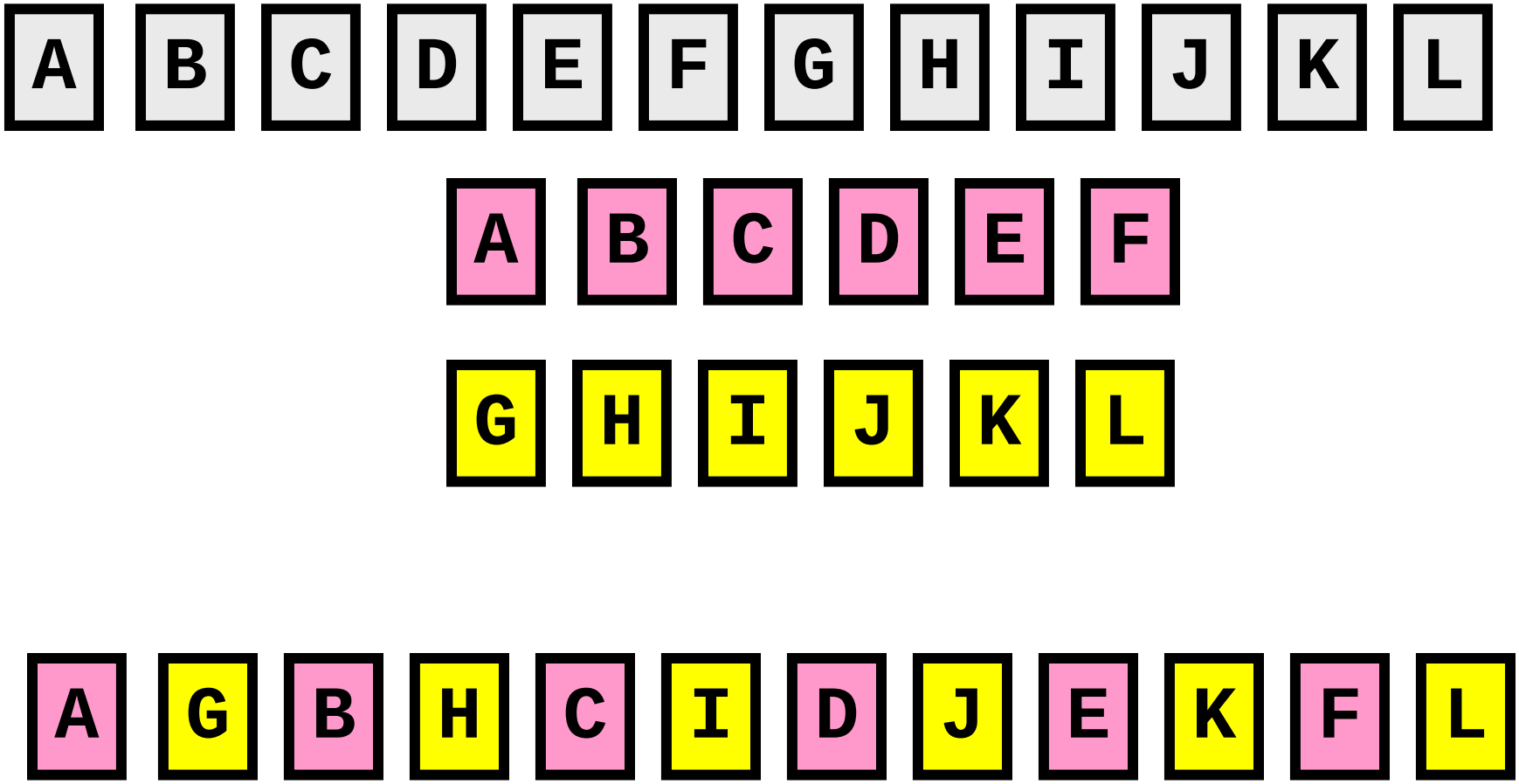
Shuffle a length-12 Card Deck



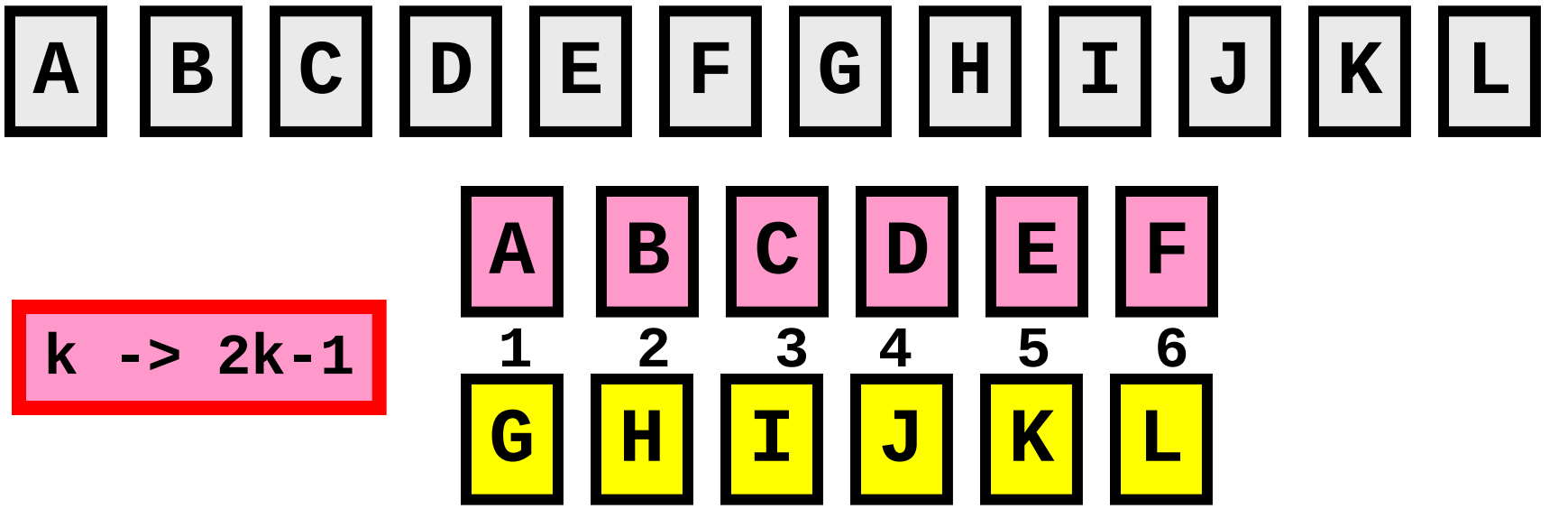
Step 1: Cut the Deck



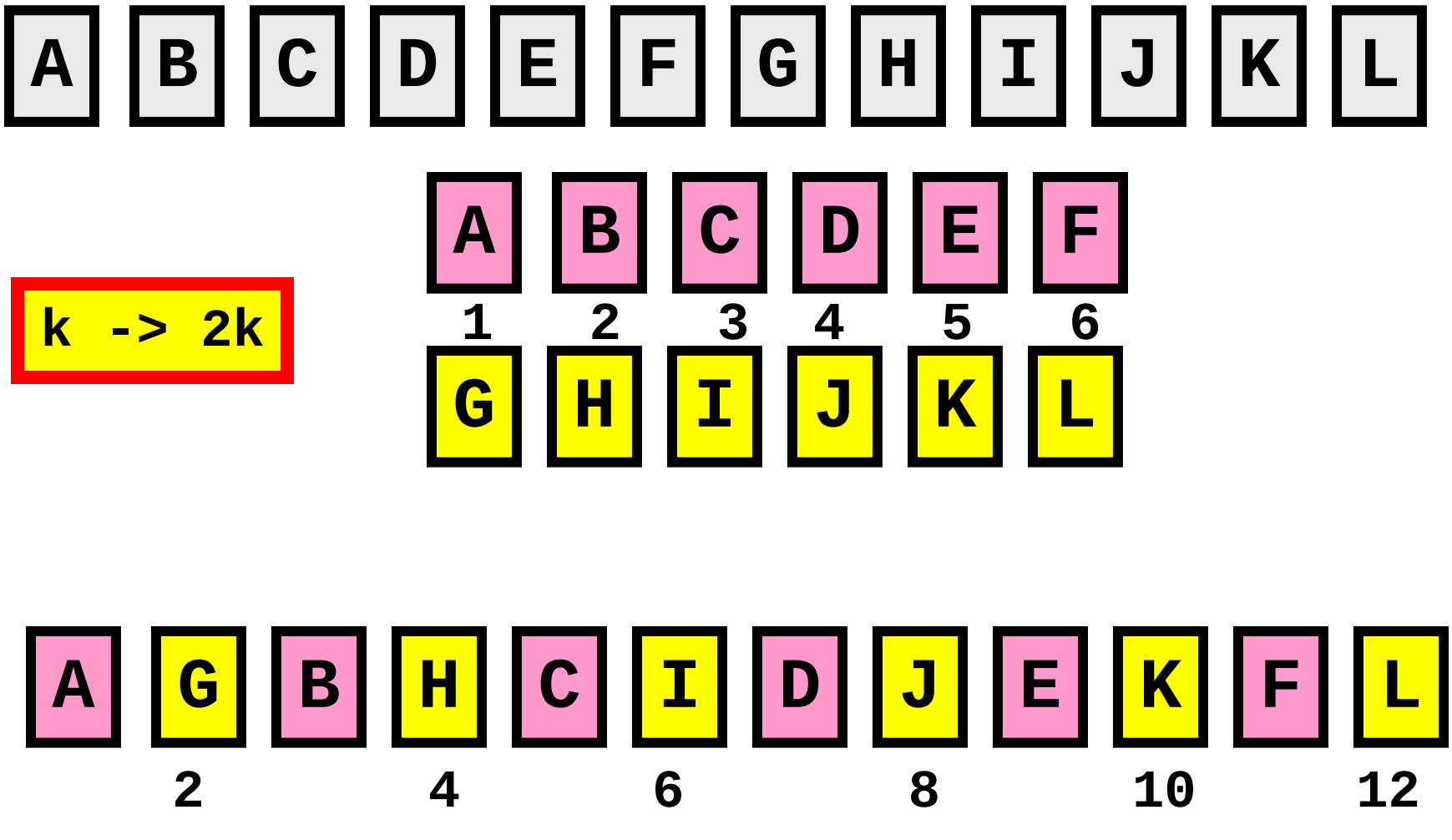
Step 2: Alternate



Step 2: Alternate



Step 2: Alternate



Resulting Code

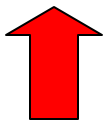
```
function T = Shuffle(S)
n = length(S); m = n/2;
T = cell(n,1);
Top = S(1:m); Bot = S(m+1:n);
for k=1:m
   T{2*k-1} = Top{k};
   T{2*k}   = Bot{k};
end
```

8 Shuffles with a Card Deck…

And you are back where you started

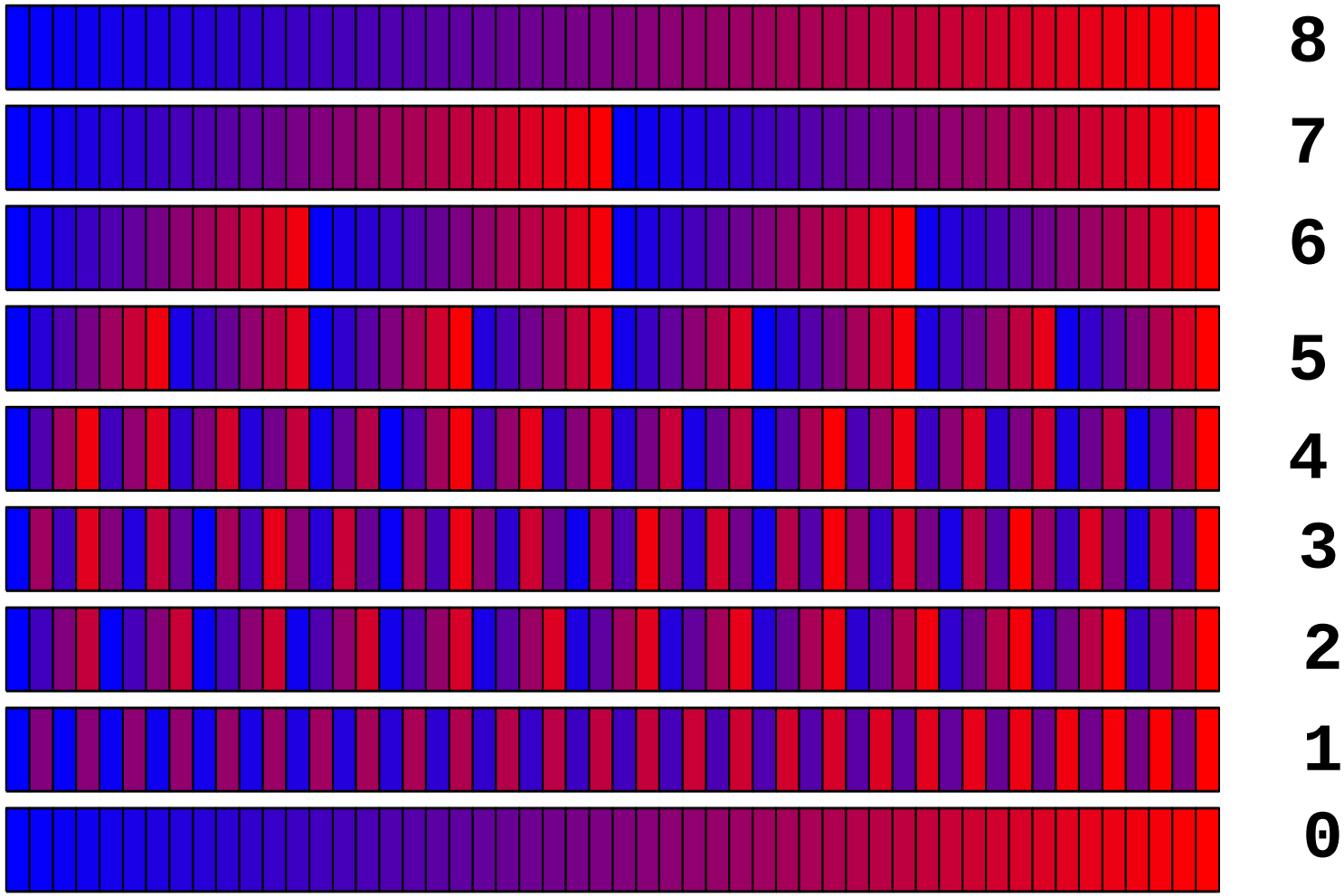
Illustrate with Color

```
% Set up a 52-color spectrum

C = cell(52,1);
for k = 1:52
     f = (k-1)/51;
     C{k} = [f 0 1-f];
end
```



These are colors

Using `fill( , , C{k})`…



If You Want a Random Shuffle…

- Use built-in function: randperm(n)
  - Produces a random permutation of the numbers 1:n

```
>> randperm(52)
ans =

  Columns 1 through 15
    24    31     8    42    21    25    15    50    51    27    30    39    26     2    29

  Columns 16 through 30
    49    22    44    16    19    36    48    10    33     7    35     4    46    38    28

  Columns 31 through 45
     3    11    40    43    52    47    14    32     6    12    23     9    45    41    37

  Columns 46 through 52
     5    20    18    13    34    17     1
```

Problem:
Build Cell Array of Roman Numerals

Goal…

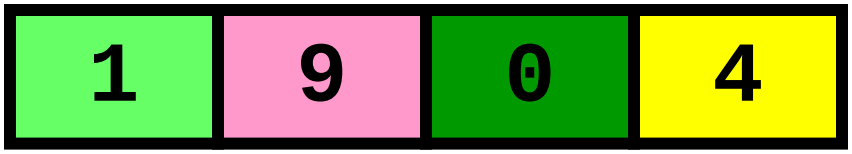
```
C{1} = ‘I’
C{2} = ‘II’
C{3} = ‘III’
   :
C{2007} = ‘MMVII’
   :
C{3999} = ‘MMMXMXCIX’
```

A Conversion Problem

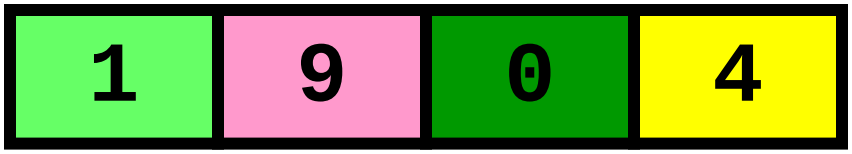
1904 = 1*1000 + 9*100 + 0*10 + 4*1

= M CM IV

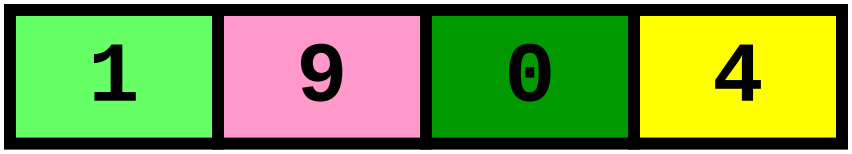
= MCMIV



MCMIV



‘M’ || ‘CM’ || ‘’ || ‘IV’



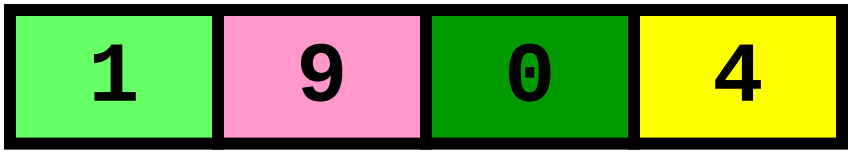
‘M’ || ‘CM’ || ‘’ || ‘IV’

‘’

M

MM

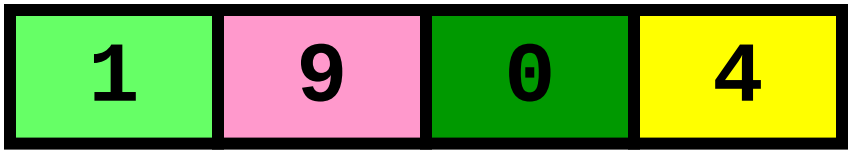
MMM



‘M’ || ‘CM’ || ‘’ || ‘IV’

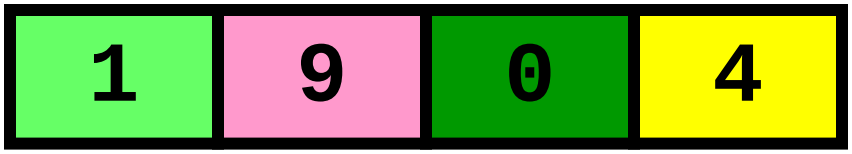
‘’
M
MM
MMM

‘’
C
CC
CCC
CD
D
DC
DCC
DCCC
CM



‘M’ || ‘CM’ || ‘’ || ‘IV’

‘’	‘’	‘’
M	C	X
MM	CC	XX
MMM	CCC	XXX
	CD	XL
	D	L
	DC	LX
	DCC	LXX
	DCCC	LXXX
	CM	XC



‘M’ || ‘CM’ || ‘’ || ‘IV’

‘’	‘’	‘’	‘’
M	C	X	I
MM	CC	XX	II
MMM	CCC	XXX	III
	CD	XL	IV
	D	L	V
	DC	LX	VI
	DCC	LXX	VII
	DCCC	LXXX	VIII
	CM	XC	IX

Concatenate entries from these cell arrays

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
```

Tens-Place Conversion

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

Tens = {'X', 'XX', 'XXX', 'XL', 'L', 'LX','LXX', 'LXXX', 'XC'};

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

Hundreds-Place Conversion

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

Hund = {'C', 'CC', 'CCC', 'CD', 'D', 'DC','DCC', 'DCCC', 'CM'};

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

Thousands-Place Conversion

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

Thou = {'M', 'MM', 'MMM'};

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

Back to Our Problem

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

This Prints 0,…,3999

```
for a = 0:3
  for b = 0:9
    for c = 0:9
      for d = 0:9

        n = a*1000 + b*100 + c*10 + d

      end
    end
  end
end
```

Reverse Problem

Given Roman Numeral, compute its value.

Assume cell array C(3999,1) available:

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

Code for Reverse Problem

```
function k = RN2Int(r)
% r is a string that represents a Roman numeral
% k is its value

  C = RomanNum;
  k=1;
  while ~strcmp(r,C{k})
     k=k+1;
  end
```