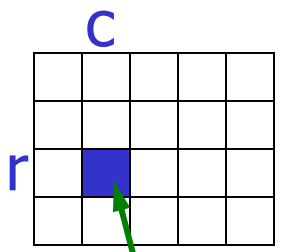
- Today's Lecture:
  - 2-d array—matrix
  - Function & subfunction
  - Details on `for`-loop

- Announcements:
  - Friday: lab session in Upson B7
  - Assignment 1b due Tuesday 11:59pm
  - Test 1 on SEP 19th in class; review on next Tuesday.

2-d array: **matrix**



- An array is a named collection of like data organized into rows and columns
- A 2-d array is a table, called a *matrix*
- Two *indices* identify the position of a value in a matrix, e.g.,

  `mat(r,c)`

  refers to component in row r, column c of matrix mat
- Array index starts at 1
- Rectangular: all rows have the same #of columns

Creating a matrix

- Built-in functions: ones, zeros, rand
  - E.g., zeros(2,3) gives a 2-by-3 matrix of 0s
- "Build" a matrix using square brackets, [ ], but the dimension must match up:
  - [x y] puts y to the right of x
  - [x; y] puts y below x
  - [4 0 3; 5 1 9] creates the matrix

4	0	3
5	1	9

  - [4 0 3; ones(1,3)] gives

4	0	3
1	1	1

  - [4 0 3; ones(3,1)] doesn't work

Working with a matrix:
`size` and individual components

Given a matrix M

2	-1	.5	0	-3
3	8	6	7	7
5	-3	8.5	9	10
52	81	.5	7	2

```
[nr, nc]= size(M)   % nr is #of rows,
                    % nc is #of columns
nr= size(M, 1)   % # of rows
nc= size(M, 2)   % # of columns

M(2,4)= 1;
disp(M(3,1))
M(1,nc)= 4;
```

```
% What will M be?
M = [ones(1,3); 1:4]
```

A

```
1   1   1   0
1   2   3   4
```

B

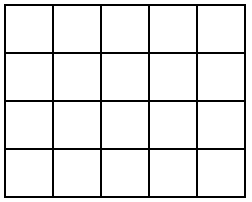
```
1   1   1
1   2   3
```

C

Error – M not created

```
% Given an nr-by-nc matrix M.
% What is A?
for  r= 1: nr
    for c= 1: nc
        A(c,r)= M(r,c);
    end
end
```

Example: minimum value in a matrix



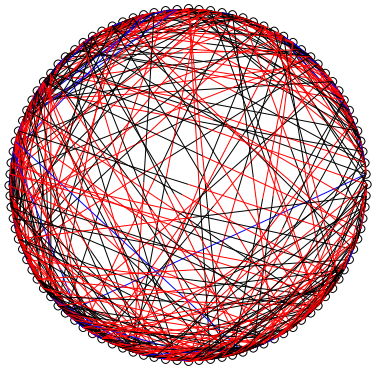
```
function val = minInMatrix(M)
% val is the smallest value in matrix M
```

```
function val = minInMatrix(M)
% val is the smallest value in matrix M

val= M(1,1);
[nr, nc]= size(M);

for r= 1:nr
    % Scan row r
    for c= 1:nc
        if  M(r,c)<val
            val= M(r,c);
        end
    end
end
```

Pattern for traversing a matrix M

```
[nr, nc] = size(M)
for  r= 1:nr
      % At row r
      for  c= 1:nc
            % At column c (in row r)
            %
            % Do something with M(r,c) …
      end
end
```

Matrix example: Random Web



- N web pages can be represented by an N-by-N Link Array A.
- A(i,j) is 1 if there is a link on webpage j to webpage i
- Generate a random link array and display the connectivity:
  - There is no link from a page to itself
  - If i≠j then A(i,j) = 1 with probability $\frac{1}{1+|i-j|}$

  ⇨ There is more likely to be a link if i is close to j

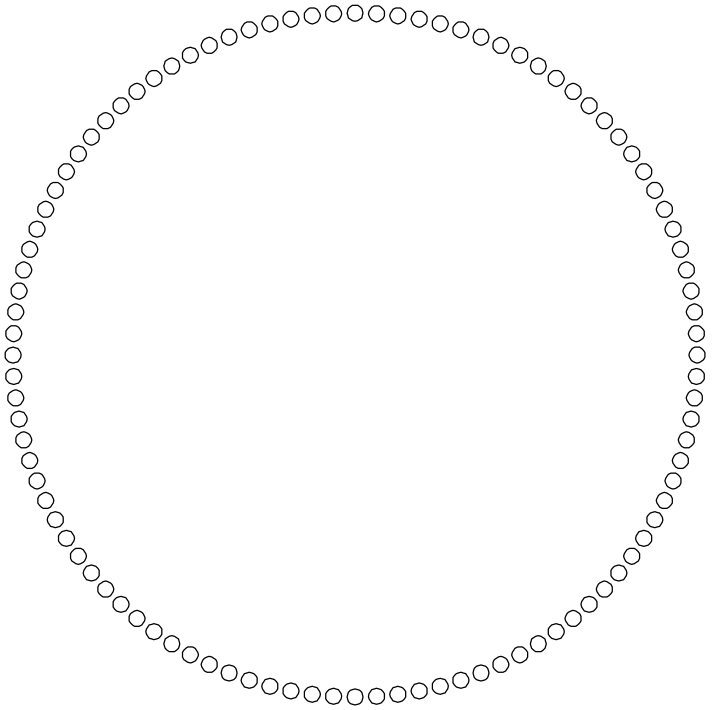
```
function A = RandomLinks(n)
% A is n-by-n matrix of 1s and 0s
% representing n webpages

A = zeros(n,n);
for i=1:n
   for j=1:n
      r = rand(1);
      if i~=j && r<= 1/(1 + abs(i-j));
          A(i,j) = 1;
      end
   end
end
```

Random web

N = 20

```
0 1 1 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0
1 0 0 0 1 0 0 0 1 1 1 0 0 0 0 0 1 0 0
0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0
0 1 1 1 1 1 0 0 0 1 0 1 1 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1 1
0 1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 1
0 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 0
0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1
0 0 0 1 0 0 0 0 1 1 0 1 0 1 1 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0
0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 1 0 0 0 1
0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 1 0
0 1 0 0 0 0 0 0 1 0 0 0 0 1 0 1 0 1 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0
```

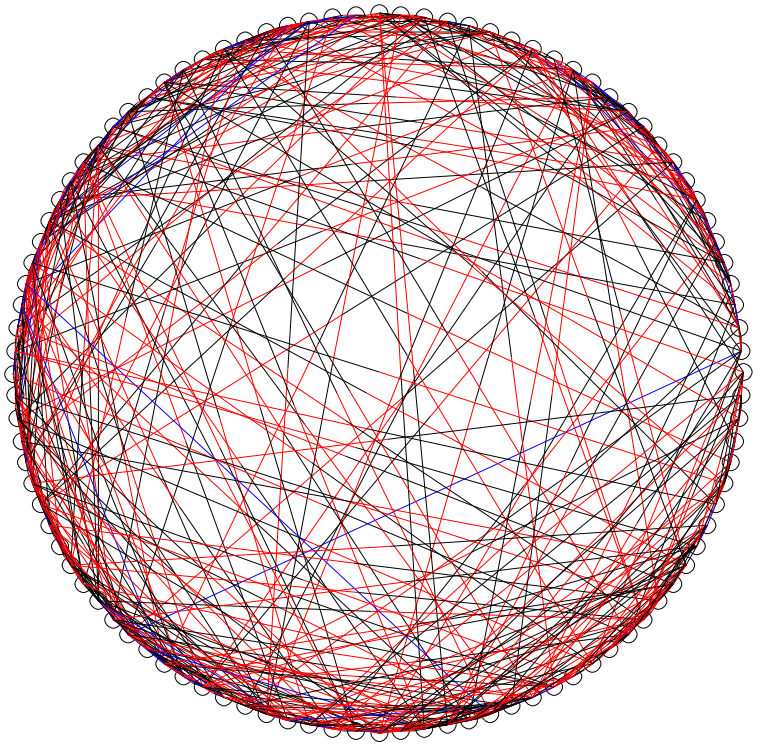
Represent the web pages graphically…



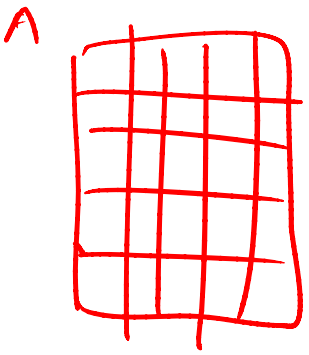
100 Web pages arranged in a circle.
Next display the links….

Represent the web pages graphically…

See
`ShowRandomLinks.m`



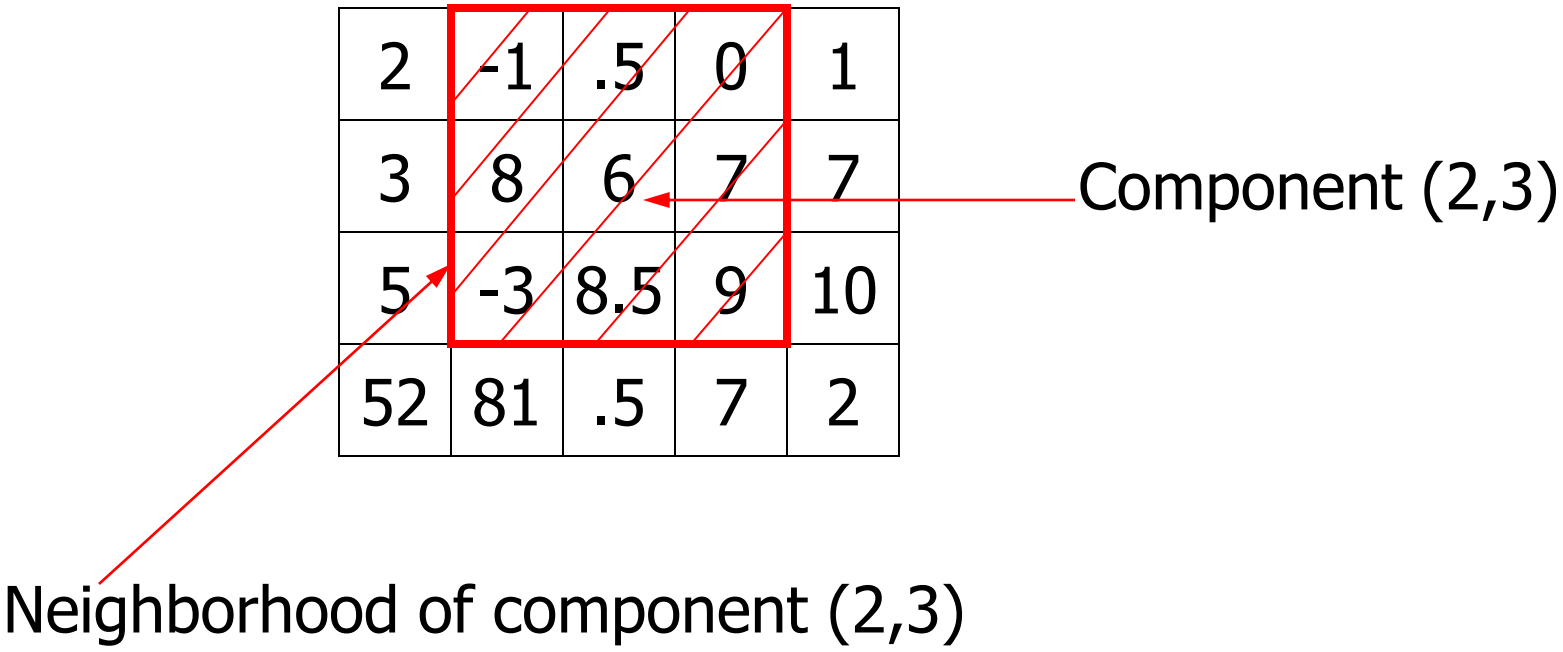
Bidirectional links are blue. Unidirectional link is black as it leaves page j, red when it arrives at page i.



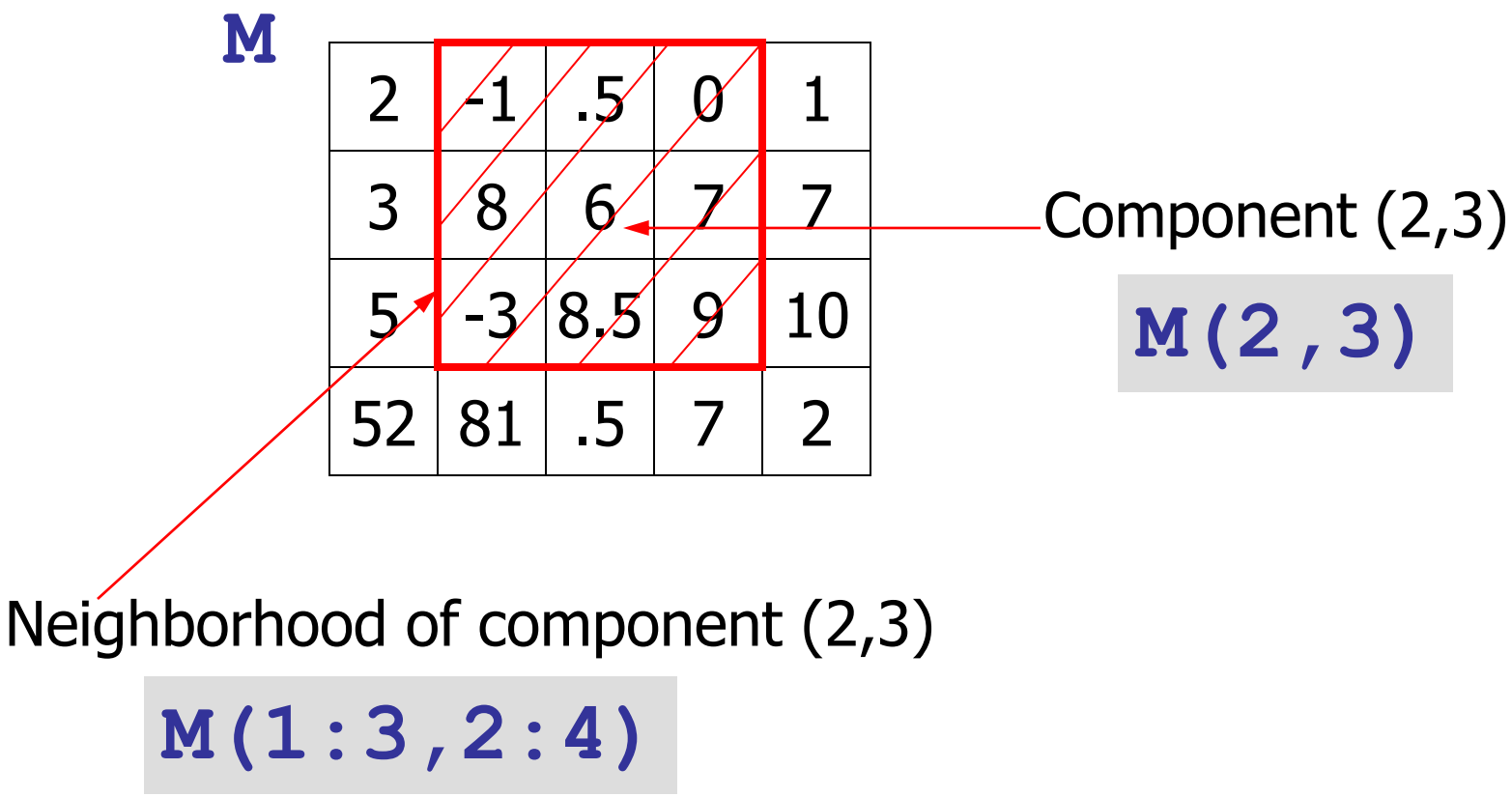
```
for i = 1:n
    for j = 1:n
        if A(i,j) == 1 && A(j,i) == 1
            % Blue
        elseif A(i,j) == 1
            % Black - Red
            % j -> mid   mid -> i
        end
    end
end
```

```
n = 10;  % No. of web pages
A = RandomLinks(n);
close all
figure
axis equal square off
hold on

% n webpages laid out around a circle
[x, y] = CirclePoints(n);
plot(x, y, 'ko')
% Show the page index on the diagram:
for k= 1:length(x)
   text(x(k),y(k), sprintf('  %d', k))
end

% Use link matrix A to connect appropriate pages (points)
for i= 1:n
    for j= i+1:n
        if A(i,j)==1 && A(j,i)==1  % bidirectional link
            plot([x(j) x(i)], [y(j) y(i)], 'b')
        elseif A(i,j)==1  % link to page i from page j
            xmid = (x(i) + x(j)) / 2;
            ymid = (y(i) + y(j)) / 2;
            plot ([x(j), xmid], [y(j), ymid], 'k')
            plot ([xmid, x(i)], [ymid, y(i)], 'r')
        elseif A(j,i)==1  % link to page j from page i
            xmid = (x(i) + x(j)) / 2;
            ymid = (y(i) + y(j)) / 2;
            plot ([x(i), xmid], [y(i), ymid], 'k')
            plot ([xmid, x(j)], [ymid, y(j)], 'r')
        end
    end
end
```

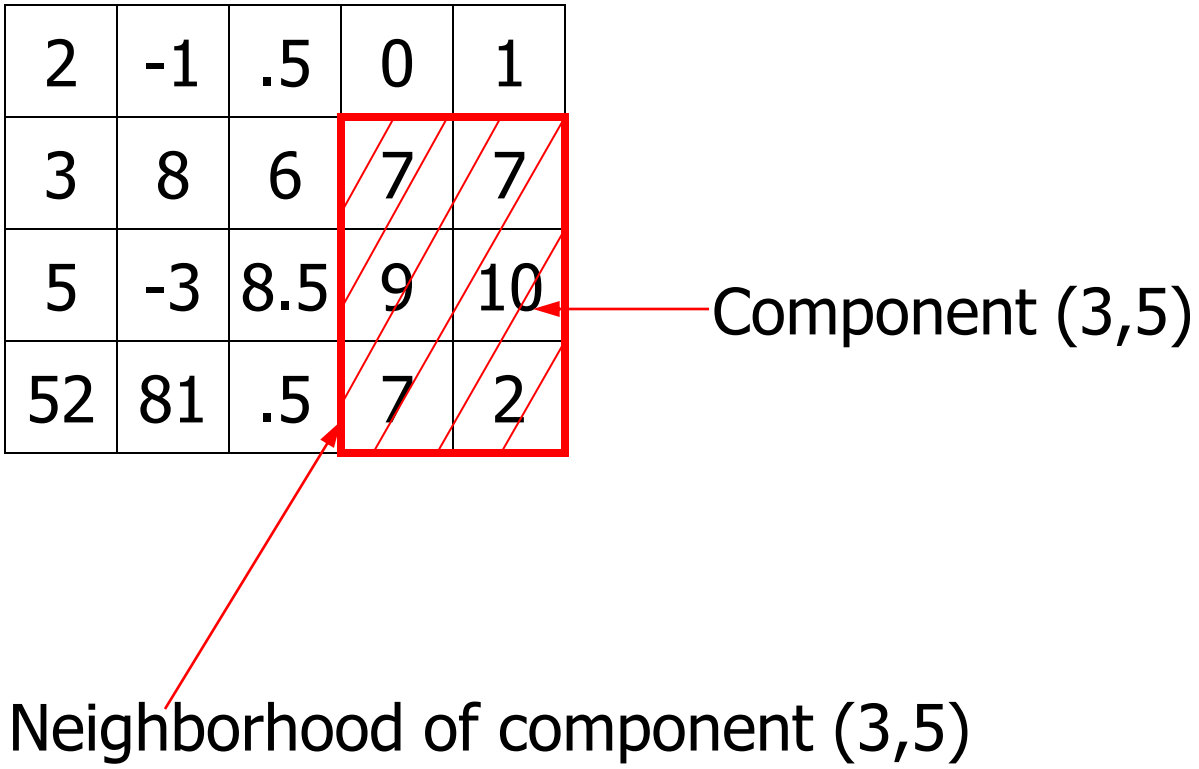
Local minimum in a neighborhood



Accessing a submatrix



Local minimum in a neighborhood



Local minimum in a neighborhood

- Write a function minInNeighborhood
- Input parameters:
  - M: matrix of numeric values
  - loc: location of the middle of the neighborhood

    loc(1), loc(2) are the row, column numbers
- Output parameter: minVal

  The minimum value of the neighborhood

Ask yourself questions!

- Can you find the min of a (sub)matrix?
  - Yes! Our function minInMatrix(A)
- Given the indices r, c (representing element M(r,c)), is it easy to define the neighborhood?
  - Yes, for the general case the neighborhood is

    M(r-1:r+1, c-1:c+1)
  - But need to deal with the "border cases"

Local minimum in a neighborhood

M

2	-1	.5	0	1
3	8	6	7	7
5	-3	8.5	9	10
52	81	.5	7	2

Component (3,5)

Want to be able to use the general case,

```
M(r-1:r+1,c-1:c+1)
```

Local minimum in a neighborhood

M

2	-1	.5	0	1
3	8	6	7	7
5	-3	8.5	9	10
52	81	.5	7	2

Want to be able to use the general case,

```
m(r-1:r+1,c-1:c+1)
```

Local minimum in a neighborhood

B	B	B	B	B	B	B
B	2	-1	.5	0	1	B
B	3	8	6	7	7	B
B	5	-3	8.5	9	10	B
B	52	81	.5	7	2	B
B	B	B	B	B	B	B

Create a border of values (B is some big number)

Want to be able to use the general case,

```
m(r-1:r+1,c-1:c+1)
```

Note: This is an exercise on manipulating a matrix. Method not suitable for a large matrix!

```
function minVal = minInNeighborhoodV3(M, loc)
% minVal is the lowest value in the neighborhood of position
% (loc(1),loc(2)), i.e., M(loc(1)-1:loc(1)+1, loc(2)-1:loc(2)+1).
% M is a matrix of numbers.  loc is a vector of length 2.

[nr, nc]= size(M);  % dimensions of M

% Build a border around M using NON-VECTORIZED code
bigM= zeros(nr+2,nc+2);
for r= 1:nr+2
    for c= 1:nc+2;
        if r==1 || r==nr+2 || c==1 ||c==nc+2
            bigM(r,c)= realmax;
        else
            bigM(r,c)= M(r-1,c-1);
        end
    end
end

nr= nr+2;  nc= nc+2;          % dimensions of bigM
r= loc(1)+1;  c= loc(2)+1;  % center of neighborhood in bigM

minVal= minInMatrix( bigM(r-1:r+1, c-1:c+1) );
```

```
function minVal = minInNeighborhoodV2(M, loc)
% minVal is the lowest value in the neighborhood of position
% (loc(1),loc(2)), i.e., M(loc(1)-1:loc(1)+1, loc(2)-1:loc(2)+1).
% M is a matrix of numbers.  loc is a vector of length 2.

[nr, nc]= size(M);  % dimensions of M

% Build a border around M by assigning M into bigM .
% This is vectorized code.
bigM= ones(nr+2,nc+2)*realmax;
bigM(2:nr+1,2:nc+1)= M;

nr= nr+2;  nc= nc+2;          % dimensions of bigM
r= loc(1)+1;  c= loc(2)+1;  % center of neighborhood in bigM

minVal= minInMatrix( bigM(r-1:r+1, c-1:c+1) );
```

Subfunction

- There can be more than one function in an M-file
- top function is the main function and has the name of the file
- remaining functions are subfunctions, accessible only by the functions in the same m-file
- Each (sub)function in the file begins with a function header
- Keyword **`end`** is not necessary at the end of a (sub)function

What will be displayed when you run the following script?

```
for k = 4:6
    disp(k)
    k= 9;
    disp(k)
end
```

```
4
9
```

A

or

```
4
4
```

B

or

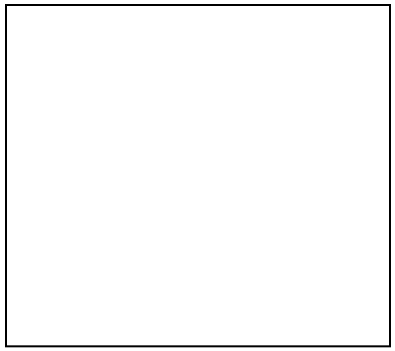
Something else …

C

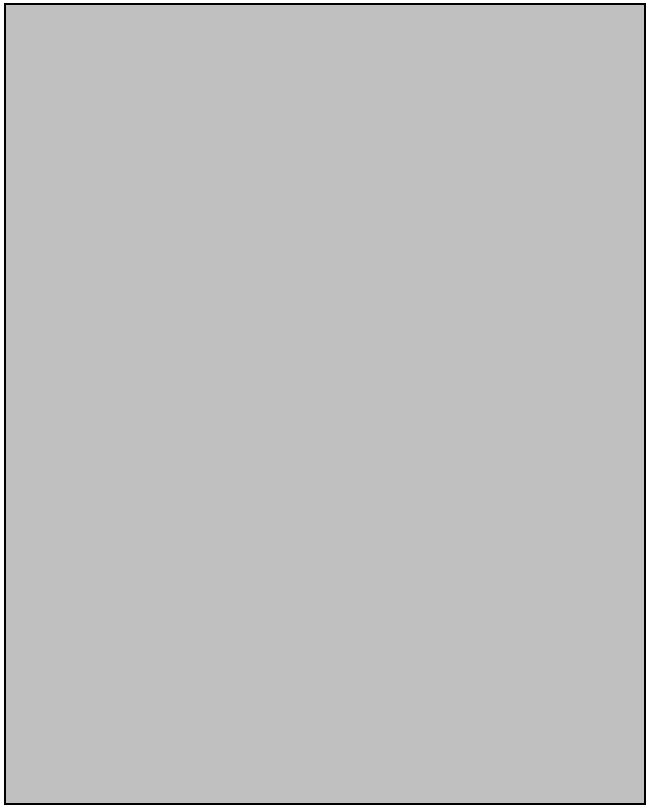
```
for k = 4:6
    disp(k)
    k= 9;
    disp(k)
end
```

With this loop header, **k** "promises" to be these values, one at a time

k



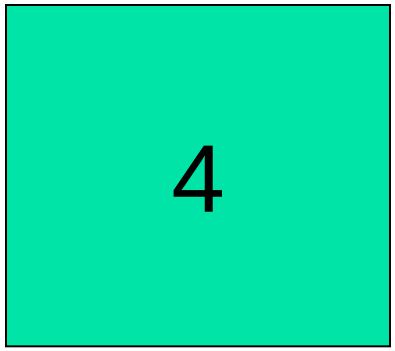
Output in Command Window



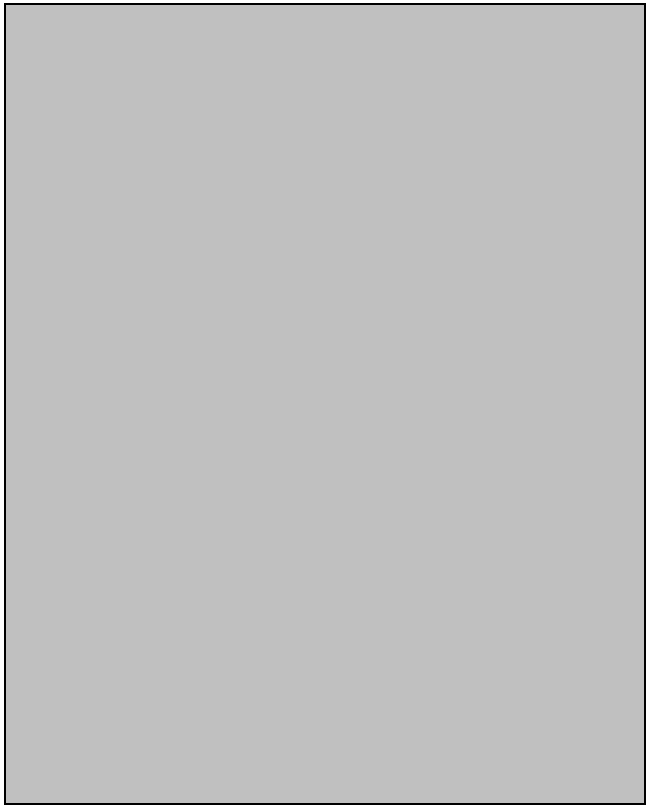
```
for k = 4:6
    disp(k)
    k= 9;
    disp(k)
end
```

With this loop header, **k** "promises" to be these values, one at a time

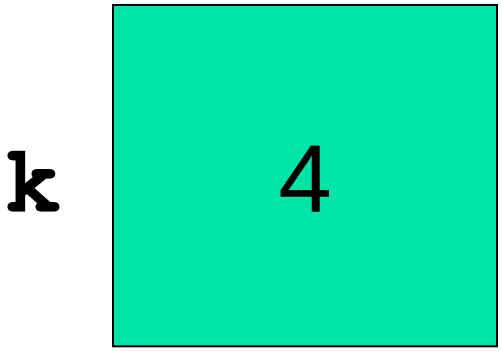
k



Output in Command Window



```
for k = 4:6
    disp(k)
    k= 9;
    disp(k)
end
```

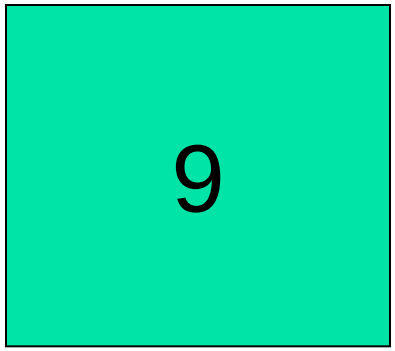


Output in Command Window

```
4
```

```
for k = 4:6
    disp(k)
    k= 9;
    disp(k)
end
```

k



Output in Command Window

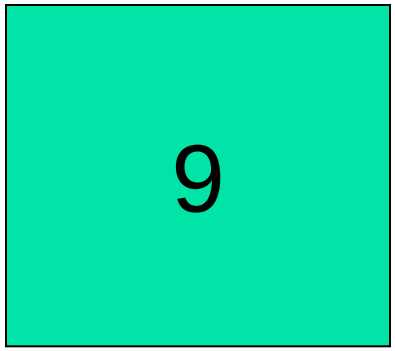
```
4
```

```
for k = 4:6
    disp(k)
    k= 9;
    disp(k)
end
```

Output in Command Window

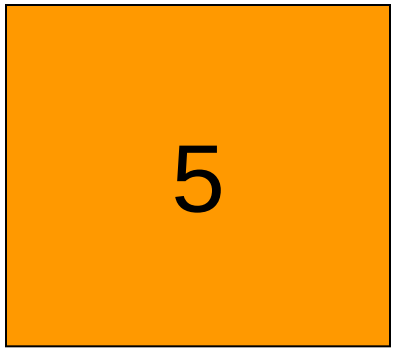
```
4
9
```

k



```
for k = 4:6
    disp(k)
    k= 9;
    disp(k)
end
```

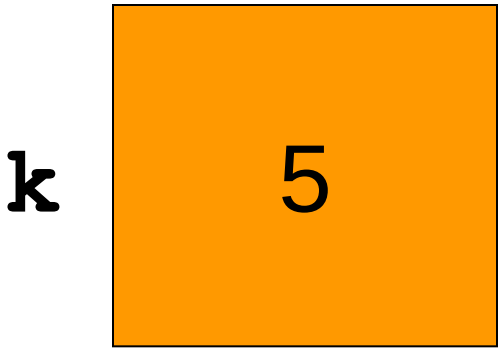
k



Output in Command Window

```
4
9
```

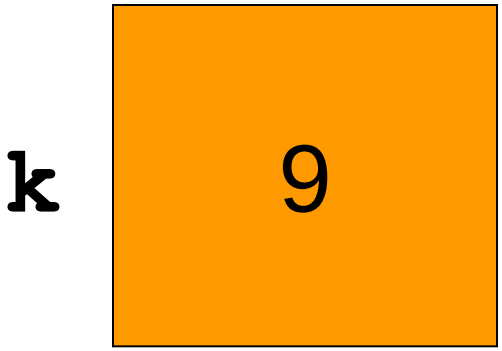
```
for k = 4:6
    disp(k)
    k= 9;
    disp(k)
end
```



Output in Command Window

```
4
9
5
```

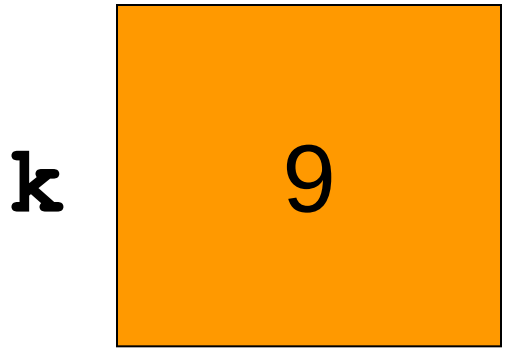
```
for k = 4:6
    disp(k)
    k= 9;
    disp(k)
end
```



Output in Command Window

```
4
9
5
```

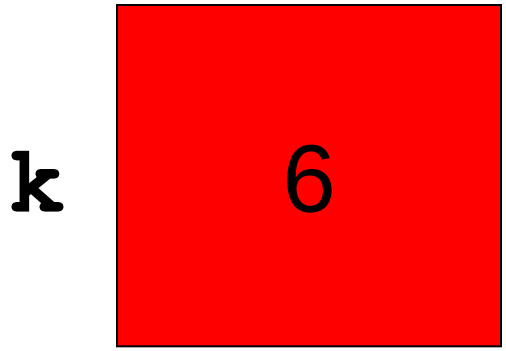
```
for k = 4:6
    disp(k)
    k= 9;
    disp(k)
end
```



Output in Command Window

```
4
9
5
9
```

```
for k = 4:6
    disp(k)
    k= 9;
    disp(k)
end
```

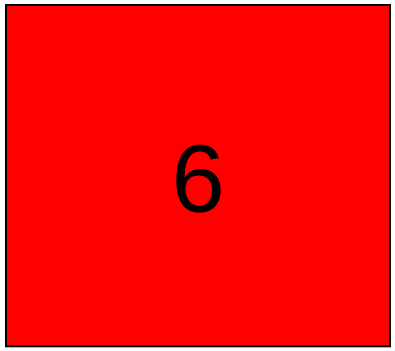


Output in Command Window

```
4
9
5
9
```

```
for k = 4:6
    disp(k)
    k= 9;
    disp(k)
end
```

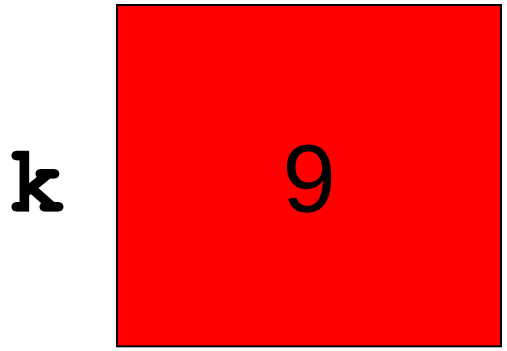
k



Output in Command Window

```
4
9
5
9
6
```

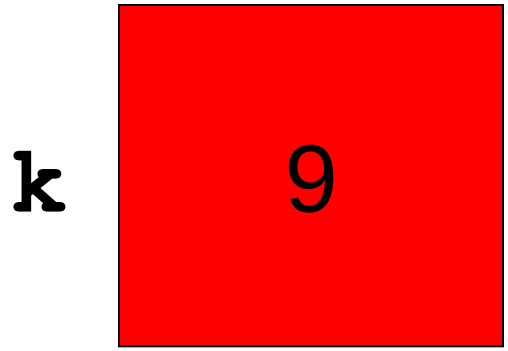
```
for k = 4:6
    disp(k)
    k= 9;
    disp(k)
end
```



Output in Command Window

```
4
9
5
9
6
```

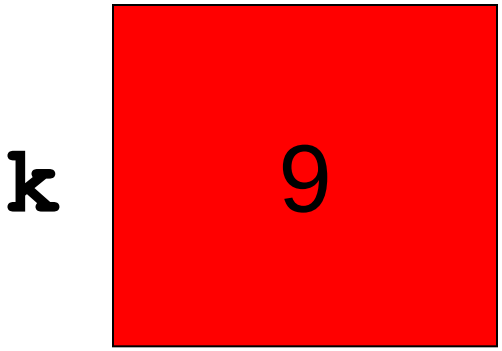
```
for k = 4:6
    disp(k)
    k= 9;
    disp(k)
end
```



Output in Command Window

```
4
9
5
9
6
9
```

```
for k = 4:6
    disp(k)
    k= 9;
    disp(k)
end
```



Output in Command Window

```
4
9
5
9
6
9
```

```
for k = 4:6
    disp(k)
    k= 9;
    disp(k)
end
```

Not a condition (boolean expression) that checks whether k<=6.

It is an expression that specifies values: