- Previous Lecture:
  - Recursion

- Today's Lecture:
  - Sorting and searching
    - Insertion sort, linear search
    - Read about *Bubble Sort* in Insight
  - "Divide and conquer" strategies
    - Binary search, merge sort

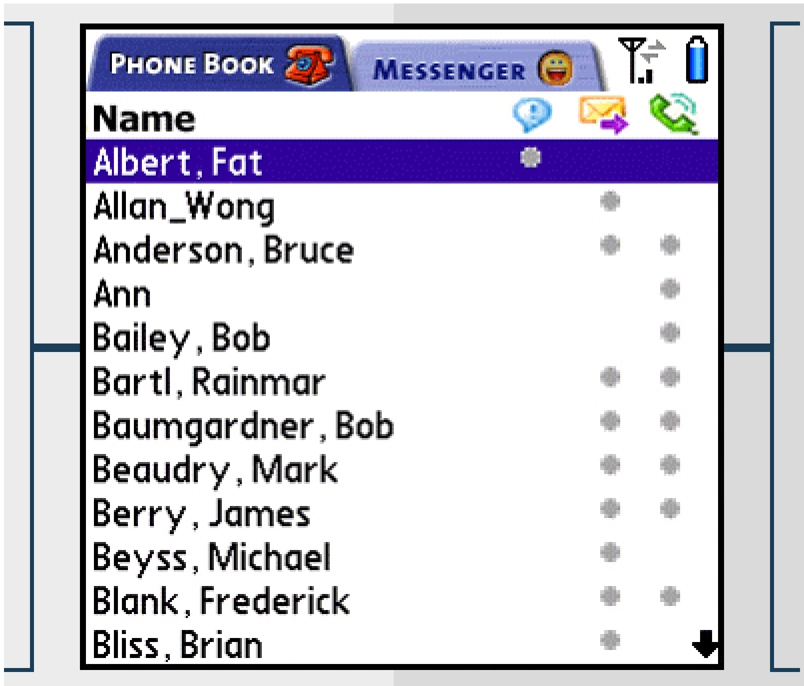
- Announcements
  - Discussion in Upson B7 lab this week
  - P6 due Thursday at 11pm
  - Final exam: Dec 17th 7pm, Barton Indoor Track WEST

Searching for an item in a collection

Is the collection organized?
What is the organizing scheme?

Indiana Jones and the Raiders of the Lost Ark

Sorting data allows us to search more easily



Boston Marathon Top Women Finishers

			Official Time	State	Country	Ctz
			2:25:25		ETH	
			2:25:27		RUS	
			2:26:34		KEN	
			2:28:12		LAT	
			2:29:48		ETH	
			2:30:52		ITA	
7	F12	Olaru, Nuta	2:33:56		ROM	
8	F6	Guta, Robe Tola	2:34:37		ETH	
9	F1	Grigoryeva, Lidiya	2:35:37		RUS	
	F35	Hood, Stephanie A.	2:44:44	IL	USA	CAN
	F14	Robson, Denise C.	2:45:54	NS	CAN	
	F11	Chemjor, Magdaline	2:46:25		KEN	
	F101	Sultanova-Zhdanova, Firaya	2:47:17	FL	USA	RUS
	F15	Mayger, Eliza M.	2:47:36		AUS	
	F24	Anklam, Ashley A.	2:48:43	MN	USA	

Name	Score	Grade
Jorge	92.1	
Ahn	91.5	
Oluban	90.6	
Chi	88.9	
Minale	88.1	
Roll	87.3	

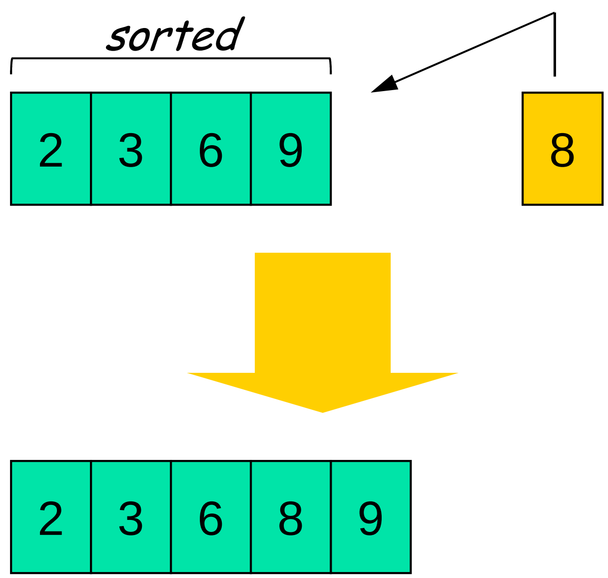
There are many algorithms for sorting

- Insertion Sort (to be discussed today)
- Bubble Sort (read *Insight* §8.2)
- Merge Sort (to be discussed Thursday)
- Quick Sort (a variant used by Matlab's built-in `sort` function)

- Each has advantages and disadvantages. Some algorithms are faster (time-efficient) while others are memory-efficient
- *Great opportunity for learning how to analyze programs and algorithms!*

The Insertion Process

- Given a sorted array x, insert a number y such that the result is sorted



Insertion

one insert process

sorted

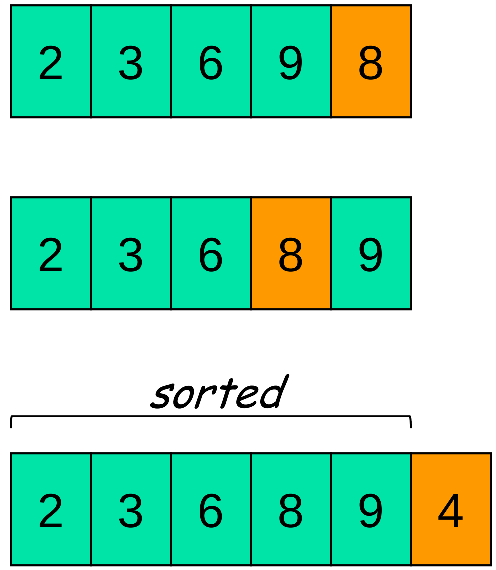
2	3	6	9	8

Insert 8 into the sorted segment

2	3	6	8	9

Just swap 8 & 9

Insertion



Insert 4 into the sorted segment

Insertion

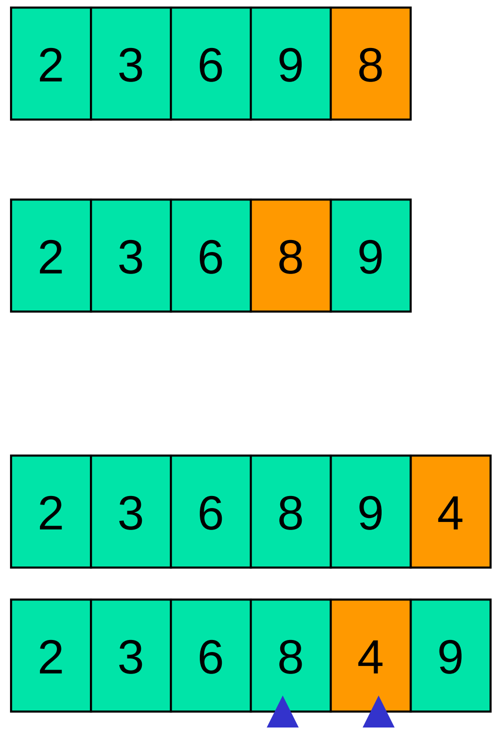
2	3	6	9	8

2	3	6	8	9

2	3	6	8	9	4

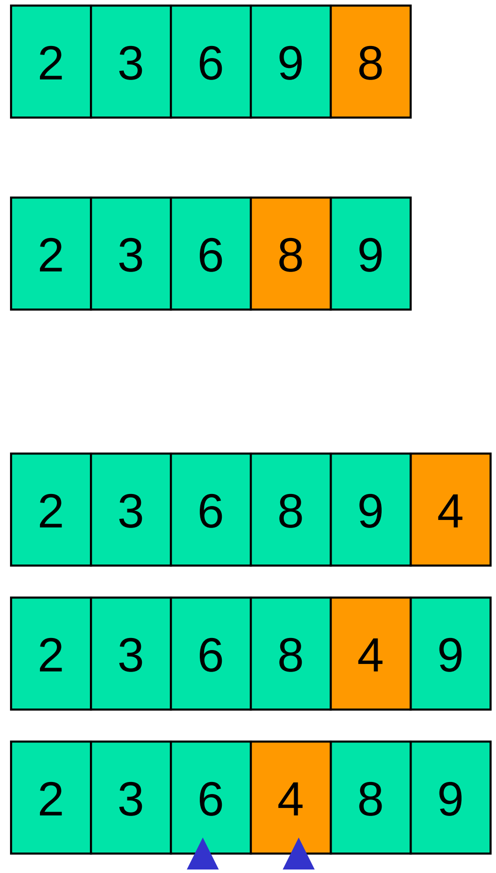
Compare adjacent components:
swap 9 & 4

Insertion



Compare adjacent components:
swap 8 & 4

Insertion



Compare adjacent components:
swap 6 & 4

Insertion

one insert process

2	3	6	9	8

2	3	6	8	9

one insert process

2	3	6	8	9	4

2	3	6	8	4	9

2	3	6	4	8	9

2	3	4	6	8	9

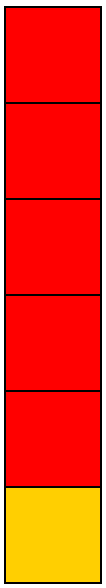
Compare adjacent components:
DONE! No more swaps.

See **`Insert.m`** for the insert process

Sort vector **x** using the Insertion Sort algorithm

Need to start with a *sorted* subvector. How do you find one?

x



Length 1 subvector is “sorted”

Insert **`x(2)`**: **`[x(1:2),C,S] = Insert(x(1:2))`**

Insert **`x(3)`**: **`[x(1:3),C,S] = Insert(x(1:3))`**

Insert **`x(4)`**: **`[x(1:4),C,S] = Insert(x(1:4))`**

Insert **`x(5)`**: **`[x(1:5),C,S] = Insert(x(1:5))`**

Insert **`x(6)`**: **`[x(1:6),C,S] = Insert(x(1:6))`**

`InsertionSort.m`

Insertion Sort vs. Bubble Sort

- Read about Bubble Sort in *Insight* §8.2
- Both algorithms involve the repeated comparison of adjacent values and swaps
- Find out which algorithm is more efficient on average

Other efficiency considerations

- Worst case, best case, average case
- Use of subfunction incurs an "overhead"
- Memory use and access

- Example: Rather than directing the *insert* process to a subfunction, have it done "in-line."
- Also, Insertion sort can be done "in-place," i.e., using "only" the memory space of the original vector.

```
function x = InsertionSortInplace(x)
% Sort vector x in ascending order with insertion sort

n = length(x);
for i= 1:n-1
      % Sort x(1:i+1) given that x(1:i) is sorted











end
```

```
function x = InsertionSortInplace(x)
% Sort vector x in ascending order with insertion sort

n = length(x);
for i= 1:n-1
      % Sort x(1:i+1) given that x(1:i) is sorted
      j= i;

      while

          % swap x(j+1) and x(j)




          j= j-1;

      end
end
```

Sort an array of objects

- Given x, a 1-d array of Interval references, sort x according to the widths of the Intervals from narrowest to widest
- Use the insertion sort algorithm
- How much of our code needs to be changed?

A. No change

B. One statement

C. About half the code

D. Most of the code

Sort an array of objects

- Given x, a 1-d array of Interval references, sort x according to the widths of the Intervals from narrowest to widest
- Use the insertion sort algorithm
- How much of our code needs to be changed?

A. No change

B. One statement

C. About half the code

D. Most of the code

The only change is in how we do the **comparison**!

See `InsertionSortIntervals.m`

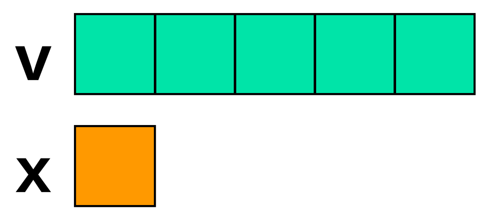
Searching for an item in a collection

Is the collection organized?
What is the organizing scheme?

Indiana Jones and the Raiders of the Lost Ark

Searching for an item in an unorganized collection?

- May need to look through the whole collection to find the target item
- E.g., find value x in vector v



- Linear search

```
% f is index of first occurrence
%   of value x in vector v.
% f is -1 if x not found.
k= 1;
while  k<=length(v) && v(k)~=x
    k= k + 1;
end
if  k>length(v)
    f= -1; % signal for x not found
else
    f= k;
end
```

```
% Linear Search
% f is index of first occurrence
%   of value x in vector v.
% f is -1 if x not found.
k= 1;
while  k<=length(v) && v(k)~=x
    k= k + 1;
end
if  k>length(v)
    f= -1; % signal for x not found
else
    f= k;
end
```

v | 12 | 35 | 33 | 15 | 42 | 45 |

x | 31 |

```
% Linear Search
% f is index of first occurrence
%   of value x in vector v.
% f is -1 if x not found.
k= 1;
while  k<=length(v) && v(k)~=x
    k= k + 1;
end
if  k>length(v)
    f= -1; % signal for x not found
else
    f= k;
end
```

A. squared

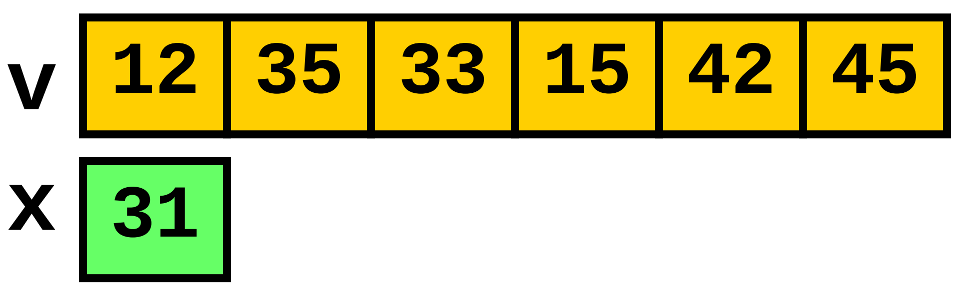
B. doubled

C. the same

D. halved

Suppose another vector is twice as long as v. The expected "effort" required to do a linear search is …

```
% Linear Search
% f is index of first occurrence
%   of value x in vector v.
% f is -1 if x not found.
k= 1;
while  k<=length(v) && v(k)~=x
    k= k + 1;
end
if  k>length(v)
    f= -1; % signal for x not found
else
    f= k;
end
```



```
% Linear Search
% f is index of first occurrence
%   of value x in vector v.
% f is -1 if x not found.
k= 1;
while  k<=length(v) && v(k)~=x
    k= k + 1;
end
if  k>length(v)
    f= -1; % signal for x no
else
    f= k;
end
```

Searching in a sorted list should require less work

v	12	15	33	35	42	45

x	31

What if **v** is sorted?

An ordered (sorted) list

The Manhattan phone book has 1,000,000+ entries.

How is it possible to locate a name by examining just a <u>tiny</u>, <u>tiny</u> fraction of those entries?

wide at SuperPages.com 195 Car C

Cartage New England Inc
26 Allen Ln Ipswich 01938.....978 356-9960
Cartagema Lydia
18 Jewett Ros 02131.....617 323-7639
Cartagena Avith
9 Bancroft Rox 02119.....617 442-9780
B Hyd 02136.....617 361-5253
Jessica 50 Decatur Cha 02129.....617 241-0152
Lucilla 174 Harvard Cam 02139.....617 491-5621
M 95 Rowe Ros 02131.....617 323-9713
Melvin 501 Green Cam 02139.....617 576-1061
Carte Nicholas
18 Appleton Boston 02116.....617 695-6996
Cartegena O 4 Milford Bos 02118.....617 338-8219
Carten Thos J Sr & Claire
1 Paradise Rd Mil 02186.....617 698-6163
Thomas & Kathleen
50 Thompson Ln Mil 02186.....617 696-6919
Carter A Ros 02131.....617 327-2257
A Roxbury.....617 442-5230
A 31 Bethune Wy Roxbury 02119.....617 442-1219
A 260 Putnam Av Cambridge 02139.....617 492-4174
A M 255 Maschsts Av Bos 02115.....617 266-7153
Adams 361 Centre St Mil 02186.....617 698-9074
Alice 108 Kilmarnock Bos 02215.....617 425-0193
Alice 45 Market Cambridge 02139.....617 945-2711
Andrew F 62 Vinal Av Som 02143.....617 625-7623
Carter Anne MD
1101 Beacon Bro 02446.....617 739-1022
Carter Athens
272 Newbury Boston 02116.....617 536-6329
B E 68 Gladeside Av Mat 02126.....617 296-6911
Carter Barbara L MD
Tufts-New England Medical Center Bos 02111
Call.....617 636-0051
Carter Becky Bos 02114.....617 523-4368
Bernard J
112 Gladstone E Bos 02128.....617 567-3430
Bithiah 25 Medway Dor 02124.....617 298-8713
Blake 26 Mt Vernon Bos 02108.....617 367-9931
Carter Broadcasting Co
20 Park Plz Bos 02116.....617 423-0210
Carter & Burgess Consultants Inc
23 East St Cam 02141.....617 225-0200
Carter C 2000 Commwlth Av Bri 02135.....617 782-2118
C 228 Faywood Av East Boston 02128.....617 569-1545
C 359 Harvard Cam 02138.....617 491-4822
C 610 Walk Hill Mat 02126.....617 296-6392
C & M 43 Burroughs Jam 02130.....617 524-9558

Carter F 24 Hillock Ros 02131.....617 327-1105
Faye & Ricky
357 Columbus Av Bos 02116.....617 437-7331
Francis S 134 Temple W Rox 02132.. 617 323-6781
Franklin & Anne
221 Mt Auburn Cam 02138.....617 354-0798
Fred 42 Haverford Jam 02130.....617 524-3078
Fred 96 Hinckley Rd Mil 02186.....617 698-1343
G & R 8 Verdun Dor 02124.....617 436-8906
G T 27 Franklin Av Som 02145.....617 623-7121
Gayle 25 Frontenac Dor 02124.....617 825-0322
Geo S 115 Moss Hill Rd Jam 02130.....617 522-3215
George 125 Nashua Bos 02114.....617 367-9548
Carter Halliday Associate
107 S Street Bos 02111.....617 456-1689
Carter Harry F
26 Runng Brk Rd W Rox 02132.....617 325-5465
Carter Hide Co Inc
146 Summer Bos 02110.....617 542-7987
Carter Hilary 61 Harvey Cam 02140.....617 876-2750
Horace
241 Walnut Av Roxbury 02119.....617 442-5307
Howard Jr 26 Notre Dme Rox 02119. 617 445-5552
J Cam.....617 354-2688
J 15 Chatham Bro 02446.....617 232-7990
J 518 Harvard Bro 02446.....617 730-9483
J 775 Vfw Pkwy West Roxbury 02132.....617 323-5574
Carter J Jacques MD
1 Brookline Pl Bro 02446.....617 735-8787
Carter J M
1410 Columbia Rd S Bos 02127.....617 464-1040
Carter J M Ornamental Ironworks
Call.....Pembroke TelNo-617 436-5353
Carter J Veal Co
48 Newmarket Sq Rox 02118.....617 442-1775
Carter James
1573 Cambridge St Cam 02138.....617 492-1214
James 182 Fisher Av Roxbury 02120.....617 739-2193
James
37 Gold Star Rd Cambridge 02140.....617 876-8841
Jas L 14 Roseberry Rd Mat 02126.....617 361-0773
Jane 114 Adena Rd Newton 02465.....617 964-0435
Jeffrey 41 Warren Av Bos 02116.....617 426-5994
John 11 Mansfield Bri 02134.....617 987-2163
John 327 Summer Bos 02210.....617 423-4334
John 40 Westwind Rd Dor 02125.....617 282-1235
June O 329 A Summit Av Bri 02135.....617 734-6109
K 38 Browning Av Dorchester 02124.....617 265-8456
K 17 Esmond Dorchester 02121.....617 282-1593

Carter Nella E
333 Maschsts Av Bos 02115.....617 267-6483
Nicholas S F
115 Randolph Av Mil 02186.....617 698-5307
Nick 21 Fairfield Bos 02116.....617 267-5222
Nick & Debbi
196 Herrick Rd Newton 02459.....617 527-0480
Nicole.....617 698-0713
Norman G
38 Chickatawbut Dor 02122.....617 822-1203
P 94 Crestwood Pk Rox 02121.....617 427-4754
P E 501 E Sixth S Bos 02127.....617 268-4213
P L 44 Hutchings Rox 02121.....617 427-9170
P R 91 Bynner Jam 02130.....617 983-8692
Paul & Constance
114 Anawan Av W Rox 02132.....617 325-2036
Paul E 501 E Sixth St S Bos 02127.....617 268-4546
Paul M 27 Union Bri 02135.....617 787-2115
Carter Pile Driving Inc 17 Beaver Ct
Framingham 01702..... Wellesley TelNo-781 235-8488
Carter Prudence
46 Franklin Watertown 02172.....617 393-3782
Prudence
46 Franklin Watertown 02172.....617 926-7063
Reginald
106 Brunswick Dorchester 02121.....617 541-2843
Renee & Andrew
10 Walnut Bos 02108.....617 720-3765
Carter Rice Dowd
Bulkley Dunton Publishing 163 Main Wilmington 01887
Toll Free-Dial '1' & Then.....800 638-1671
Cust Svc-Industrial Prod 613 Main Wilmington
Toll Free-Dial '1' & Then.....800 619-7447
Cust Svc-Printing 613 Main Wilmington
Toll Free-Dial '1' & Then.....800 648-7447
Headquarters 613 Main Wilmington 01887
Call.....978 988-7447
Ingalls Cronin 163 Main Wilmington 01887
Toll Free-Dial '1' & Then.....800 638-1673
Carter Richard
1079 Commwlth Av Brighton 02215... 617 987-0836
Richard A 97 Mt Vernon Bos 02108... 617 566-7293
Carter Richard A MD
170 Commwlth Av Bos 02116.....617 267-0710
Carter Richard K
15 Mercer S Bos 02127.....617 268-0448
Robert L 175 Richdale Av Cam 02140. 617 864-1535
Roger 150 St Botolph Bos 02115.....617 424-6148
Roy 44 Concord Av Cam 02138.....617 491-6115
Royce 18 Seminary Cha 02129.....617 241-0418

Key idea of "phone book search": repeated halving

To find the page containing Pat Reed's number…

```
while  (Phone book is longer than 1 page)
        Open to the middle page.
        if  "Reed" comes before the first entry,
                Rip and throw away the 2nd half.
        else
                Rip and throw away the 1st half.
        end
end
```

What happens to the phone book length?

```
Original:      3000 pages
After 1 rip:   1500 pages
After 2 rips:   750 pages
After 3 rips:   375 pages
After 4 rips:   188 pages
After 5 rips:    94 pages
      :
After 12 rips:    1 page
```

Binary Search

Repeatedly halving the size of the "search space" is the main idea behind the method of binary search.

An item in a sorted array of length n can be located with just $\log_2 n$ comparisons.

```
% Linear Search
% f is index of first occurrence of value x in vector v.
% f is -1 if x not found.
k= 1;
while  k<=length(v) && v(k)~=x
    k= k + 1;
end
if  k>length(v)
    f= -1; % signal for x not found
else
    f= k;
end
```

n comparisons against the target are needed in worst case, **n=length(v)**.

Binary Search

Repeatedly halving the size of the "search space" is the main idea behind the method of binary search.

An item in a sorted array of length n can be located with just $\log_2 n$ comparisons.

"Savings" is significant!

n	log2(n)
100	7
1000	10
10000	13

Binary search: target x = 70

	1	2	3	4	5	6	7	8	9	10	11	12
v	12	15	33	35	42	45	51	62	73	75	86	98

L: 1

Mid: 6

R: 12

`v(Mid) <= x`

So throw away the left half…

Binary search: target x = 70

	1	2	3	4	5	6	7	8	9	10	11	12
v	**12**	**15**	**33**	**35**	**42**	**45**	**51**	**62**	**73**	**75**	**86**	**98**

L: **6**

Mid: **9**

R: **12**

`x < v(Mid)`

So throw away the right half...

Binary search: target x = 70

	1	2	3	4	5	6	7	8	9	10	11	12
v	**12**	**15**	**33**	**35**	**42**	**45**	**51**	**62**	**73**	**75**	**86**	**98**

L: 6

Mid: 7

R: 9

v(Mid) <= x

So throw away the left half...

Binary search: target x = 70

	1	2	3	4	5	6	7	8	9	10	11	12
v	12	15	33	35	42	45	51	62	73	75	86	98

L: 7

Mid: 8

R: 9

v(Mid) <= x

So throw away the left half…

Binary search: target x = 70

	1	2	3	4	5	6	7	8	9	10	11	12
v	**12**	**15**	**33**	**35**	**42**	**45**	**51**	**62**	**73**	**75**	**86**	**98**

L: **8**

Mid: **8**

R: **9**

Done because

R-L = 1

```
function L = binarySearch(x, v)
% Find position after which to insert x. v(1)<…<v(end).
% L is the index such that v(L) <= x < v(L+1);
% L=0 if x<v(1).  If x>v(end), L=length(v) but x~=v(L).

% Maintain a search window [L..R] such that v(L)<=x<v(R).
% Since x may not be in v, initially set ...
L=0;  R=length(v)+1;

% Keep halving [L..R] until R-L is 1,
%   always keeping  v(L) <= x < v(R)
while  R ~= L+1
    m= floor((L+R)/2);  % middle of search window
    if


    else


    end
end
```

```
function L = binarySearch(x, v)
% Find position after which to insert x. v(1)<…<v(end).
% L is the index such that v(L) <= x < v(L+1);
% L=0 if x<v(1).  If x>v(end), L=length(v) but x~=v(L).

% Maintain a search window [L..R] such that v(L)<=x<v(R).
% Since x may not be in v, initially set ...
L=0;  R=length(v)+1;

% Keep halving [L..R] until R-L is 1,
%   always keeping  v(L) <= x < v(R)
while  R ~= L+1
    m= floor((L+R)/2);  % middle of search window
    if  v(m) <= x

        L= m;
    else

        R= m;
    end
end
```

This version is different from that in *Insight*

```
function L = binarySearch(x, v)
% Find position after which to insert x. v(1)<…<v(end).
% L is the index such that v(L) <= x < v(L+1);
% L=0 if x<v(1).  If x>v(end), L=length(v) but x~=v(L).

% Maintain a search window [L..R] such that v(L)<=x<v(R).
% Since x may not be in v, initially set ...
L=0;  R=length(v)+1;

% Keep halving [L..R] until R-L is 1,
%   always keeping  v(L) <= x < v(R)
while  R ~= L+1
    m= floor((L+R)/2);  % middle of search window
    if  v(m) <= x

        L= m;
    else

        R= m;
    end
end
```

20	30	40	46	50	52	68	70

0 1 2 3 4 5 6 7 8 9

Play with **`showBinarySearch.m`**