

- Previous Lecture (and Discussion):
 - Branching (`if`, `elseif`, `else`, `end`)
 - Relational operators (`<`, `>`, `=`, `~=`, `...`, etc.)
 - Logical operators (`&&`, `||`, `~`)
- Today's Lecture:
 - Logical operators and "short-circuiting"
 - More branching—*nesting*
 - Top-down design
- Announcements:
 - Discussion this week in Upson B7 computer lab
 - Project 1 (P1) due Thursday at 11pm
 - Submit `.m` files (plain text, not from a word processing software such as Microsoft Word)
 - Register your clicker using the link on the course website

Lecture 4 1

Consider the quadratic function

$$q(x) = x^2 + bx + c$$

on the interval $[L, R]$:

- Is the function strictly increasing in $[L, R]$?
- Which is **smaller**, $q(L)$ or $q(R)$?
- What is the **minimum value** of $q(x)$ in $[L, R]$?

Lecture 4 3

Modified Problem 3

Write a code fragment that prints "yes" if x_c is in the interval and "no" if it is not.

Lecture 4 5

$q(x) = x^2 + bx + c$

$x_c = -b/2$

Lecture 4 6

So what is the requirement?

```

% Determine whether xc is in
% [L,R]
xc = -b/2;

if _____

    disp('Yes')
else
    disp('No')
end
    
```

Lecture 4 9

The value of a boolean expression is either true or false.

$$(L \leq xc) \ \&\& \ (xc \leq R)$$

This (compound) boolean expression is made up of two (simple) boolean expressions. Each has a value that is **either true or false**.

Connect boolean expressions by **boolean operators**:

and	or	not
<code>&&</code>	<code> </code>	<code>~</code>

Lecture 4 11

Logical operators

&& logical and: Are both conditions true?
 E.g., we ask “is $L \leq x_c$ and $x_c \leq R$?”
 In our code: `L <= xc && xc <= R`

|| logical or: Is at least one condition true?
 E.g., we can ask if x_c is outside of $[L, R]$,
 i.e., “is $x_c < L$ or $R < x_c$?”
 In code: `xc < L || R < xc`

~ logical not: Negation
 E.g., we can ask if x_c is **not outside** $[L, R]$.
 In code: `~(xc < L || R < xc)`

Lecture 4 15

“Truth table”

X, Y represent boolean expressions.
 E.g., `d > 3.14`

X	Y	X && Y “and”	X Y “or”	~Y “not”
F	F			
F	T			
T	F			
T	T			

Lecture 4 22

Logical operators “short-circuit”

`a > b && c > d`

true

A **&&** expression short-circuits to false if the left operand evaluates to **false**.

`a > b && c > d`

false

Entire expression is false since the first part is false

A **||** expression short-circuits to _____ if _____

Lecture 4 25

Always use logical operators to connect simple boolean expressions

Why is it wrong to use the expression
`L <= xc <= R`
 for checking if x_c is in $[L, R]$?

Example: Suppose L is 5, R is 8, and xc is 10. We know that 10 is not in $[5, 8]$, but the expression
`L <= xc <= R` gives...

Lecture 4 28

Variables a , b , and c have whole number values. **True** or **false**: This fragment prints “Yes” if there is a *right triangle* with side lengths a , b , and c and prints “No” otherwise.

```

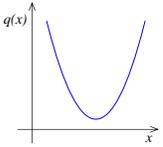
if a^2 + b^2 == c^2
    disp('Yes')
else
    disp('No')
end
    
```

A: true

B: false

Lecture 4 29

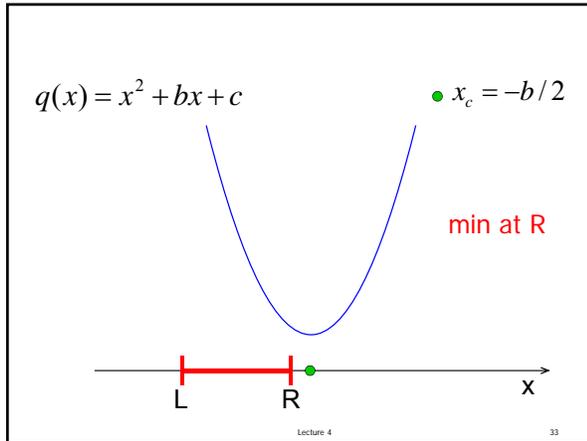
Consider the quadratic function

$$q(x) = x^2 + bx + c$$


on the interval $[L, R]$:

- Is the function strictly increasing in $[L, R]$?
- Which is **smaller**, $q(L)$ or $q(R)$?
- What is the **minimum value** of $q(x)$ in $[L, R]$?

Lecture 4 32



Start with pseudocode

If x_c is between L and R

 Min is at x_c

Otherwise

 Min is at one of the endpoints

We have decomposed the problem into three pieces! Can choose to work with any piece next: the if-else construct/condition, min at x_c , or min at an endpoint

Set up structure first: if-else, condition

```
if L<=xc && xc<=R
    Then min is at xc
else
    Min is at one of the endpoints
end
```

Now refine our solution-in-progress. I'll choose to work on the if-branch next

Refinement: filled in detail for task "min at x_c "

```
if L<=xc && xc<=R
    % min is at xc
    qMin= xc^2 + b*xc + c;
else
    Min is at one of the endpoints
end
```

Continue with refining the solution... else-branch next

Refinement: detail for task "min at an endpoint"

```
if L<=xc && xc<=R
    % min is at xc
    qMin= xc^2 + b*xc + c;
else
    % min is at one of the endpoints
    if xc < L
        % min is at L
    else % xc right of bracket
        % min is at R
    end
end
```

Continue with the refinement, i.e., replace comments with code

Final solution (given b,c,L,R,x_c)

```
if L<=xc && xc<=R
    % min is at xc
    qMin= xc^2 + b*xc + c;
else
    % min is at one of the endpoints
    if xc < L
        qMin= L^2 + b*L + c;
    else
        qMin= R^2 + b*R + c;
    end
end
```

See quadMin.m
quadMinGraph.m

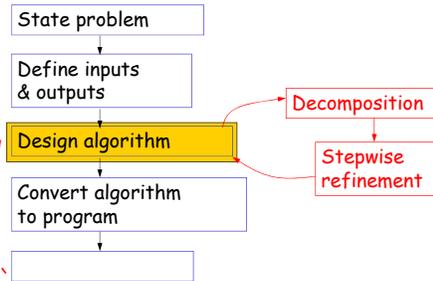
An if-statement can appear within a branch—just like any other kind of statement!

Notice that there are 3 alternatives → can use **elseif**!

```
if L<=xc && xc<=R
    % min is at xc
    qMin= xc^2+b*xc+c;
else
    % min at one endpt
    if xc < L
        qMin= L^2+b*L+c;
    else
        qMin= R^2+b*R+c;
    end
end
```

```
if L<=xc && xc<=R
    % min is at xc
    qMin= xc^2+b*xc+c;
elseif xc < L
    qMin= L^2+b*L+c;
else
    qMin= R^2+b*R+c;
end
```

Top-Down Design



An algorithm is an **idea**. To use an algorithm you must choose a programming language and **implement** the algorithm.

Does this program work?

```
score= input('Enter score: ');
if score>55
    disp('D')
elseif score>65
    disp('C')
elseif score>80
    disp('B')
elseif score>93
    disp('A')
else
    disp('Not good...')
end
```

A: yes

B: no

Question

A stick of unit length is split into two pieces. The breakpoint is randomly selected. On average, how long is the shorter piece?

- Physical experiment! ♦
- Thought experiment! → analysis
- Computational experiment! → simulation ♦

♦ Need to repeat many trials!

Simulation:

use code to imitate the physical experiment

```
% one trial of the experiment
breakPt= rand(1);
if breakPt<0.5
    shortPiece= breakPt;
else
    shortPiece= 1-breakPt;
end
```