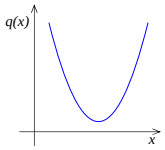
- Previous Lecture (and lab):
  - Variables & assignment
  - Built-in functions
  - Input & output
  - Good programming style (meaningful variable names; use comments)
- Today's Lecture:
  - Branching (conditional statements)

- So far, all the statements in our scripts are executed in order
- We do not have a way to specify that some statements should be executed only under some condition
- We need a new language construct…

Consider the quadratic function

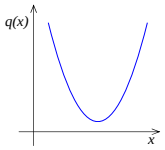
$$q(x) = x^2 + bx + c$$

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 $[L , R]$?

- What are the critical points?



Problem 1

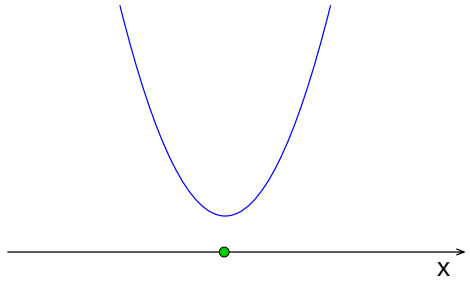
Write a code fragment that prints "yes" if q(x) increases across the interval and "no" if it does not.


```
% Quadratic q(x) = x^2 + bx + c
b = input('Enter b: ');
c = input('Enter c: ');
L = input('Enter L: ');
R = input('Enter R: ');

% Determine whether q increases
% across [L,R]
xc = -b/2;
```

The Situation

$q(x) = x^2 + bx + c$ $\quad x_c = -b/2$



So what is the requirement?

```
% Determine whether q increases
% across [L,R]
xc = -b/2;

if _____________

   fprintf('Yes\n')
else
   fprintf('No\n')
end
```

Relational Operators

<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equal to
~=	Not equal to

So what is the requirement?

```
% Determine whether q increases
% across [L,R]
xc = -b/2;

if _____________

   fprintf('Yes\n')
else
   disp('No')
end
```

```
disp('Yes')
```


Problem 2

Write a code fragment that prints
"qleft is smaller"
if q(L) is smaller than q(R).
If q(R) is smaller print
"qright is smaller."

Algorithm v0

Calculate q(L)
Calculate q(R)
If q(L) < q(R)
    print "qleft is smaller"
Otherwise
    print "qright is smaller"

Algorithm v0.1

Calculate x_c
If distance $\overline{x_c L}$ is smaller than distance $\overline{x_c R}$
    print "qleft is smaller"
Otherwise
    print "qright is smaller"

```
% Which is smaller, q(L) or q(R)?

xc= -b/2;  % x at center
if (abs(xc-L) == abs(xc-R))
   disp('qleft and qright are equal')
elseif (abs(xc-L) < abs(xc-R))
   disp('qleft is smaller')
else
   disp('qright is smaller')
end
```



```
% Which is smaller, q(L) or q(R)?

qL= L*L + b*L + c;  % q(L)
qR= R*R + b*R + c;  % q(R)
if (qL == qR)
   disp('qleft and qright are equal')
   fprintf('q value is %f\n', qL)
elseif (qL < qR)
   disp('qleft is smaller')
else
   disp('qright is smaller')
end
```


Consider the quadratic function

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

on the interval $[L, R]$:

What if you only want to know if $q(L)$ is close to $q(R)$?


```
% Is q(L) close to q(R)?

tol= 1e-4;  % tolerance
qL= L*L + b*L + c
qR= R*R + b*R + c
if (abs(qL-qR) < tol)
  disp('qleft and qright similar')
end
```

Name an important parameter and define it with a comment!

Simple `if` construct

```
if  boolean expression
    statements to execute if expression is true
else
    statements to execute if expression is false
end
```


Even simpler `if` construct

```
if  boolean expression
    statements to execute if expression is true
end
```

The `if` construct

```
if boolean expression1
   statements to execute if expression1 is true
elseif boolean expression2
   statements to execute if expression1 is false
   but expression2 is true
:
else
   statements to execute if all previous conditions
   are false
end
```

Can have any number of elseif branches but at most one else branch

Things to know about the `if` construct

- _____ branch of statements is executed
- There can be _____ `elseif` clauses
- There can be _____ `else` clause
- The `else` clause _____ in the construct
- The `else` clause _____ (boolean expression)

Modified Problem 3

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

So what is the requirement?

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

if _________________

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


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

```
(L<=xc) && (xc<=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
`&&`	`||`	`~`

Logical operators

`&&` logical and: Are both conditions true?
E.g., we ask "is $L \le x_c$ and $x_c \le R$?"
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 $x_c \le L$ or $R \le x_c$?"
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)`