**fImplies definition**:

The meaning of logic is, primarily, concentrated in one statements T->T that assumes the truth of a preposition and the truth of a logical implication. The result is the truth of a final statement. Everything else is an exercise in futility.

From a true statement, using true logical conclusions, we can arrive only to true statements. That is why "True implies True" is True.  
  
If from a true statement we derive to a false statement, the logic was wrong (that is implication is wrong). That is why "True implies False" is False.  
  
From a false statement we can derive both true and false statements using the correct logic. For instance, 2=-2 is false. Use the correct logic "if numbers are equal, their squares are equal". This is an implication "numbers are equal" IMPLIES "their squares are equal". The result of application of this logic to an incorrect equality 2=-2 is correct equality 4=4. Use another correct logic "if numbers are equal, their cubes are equal". The result of application of this logic to an incorrect equality 2=-2 is incorrect equality 8=-8. That is why both "False implies True" and "False implies False" are true.

1. If P is true and P implies Q then Q is true.  
2. If P is false and P implies Q then Q might be either true or false.  
3. If Q is false and P implies Q then P is false  
4. If Q is true and P implies Q then P might be either true or false.

5. If P is true and Q is true there may or may not be an implication from P to Q.  
6. If P is true and Q is false there may not be an implication from P to Q.  
7. If Q is true and P is true there may or may not be an implication from P to Q.  
8. If Q is true and P is false there may or may not be any implication from P to Q.

How about using examples

5. If P is true and Q is true there may or may not be an implication from P to Q.

EXAMPLE 1 - No implication

P:"Two angles of an equilateral triangle equal 60 degrees each"

Q:"Earth moves on elliptical orbit around Sun"

These are two true statements unrelated by any logical implication.

EXAMPLE 2 - There is an implication

P: the same as above

Q: "The third angle of this triangle equals to 60 degrees"

There is an implication that can be proven in different ways. For instance, using the theorem that sum of angles in a triangle is 180 degrees or another theorem that angles lying opposite to equal sides in a triangle are equal.

Similar approach with all other cases. You can give two examples in each case, one where there is no implication (unrelated statements) and another with implication.

By the way, you can use geometrical representation of implication. A=>B can be pictured as a small set of points A completely surrounded by a bigger set B. If point belongs to A, it belongs to B. If B does not surround A or partially intersects A, there is no implication.  
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Proofing Techniques:

1)

*Direct proof*:

Example: If a and b are consecutive integers then the sum a+b is odd.

2)

*Proof by contrapositive:*

Goal: P -> Q

*Technique*:

* Prove that ˥Q -> ˥P is true

Example: For any integer x, if is odd, then x is odd.

For this prove that, x is even then is even. BTW, is this the good example?

3)

*Proof using axioms*(mostly used in math):

Goal: Q

*Technique*:

1. Find axiom P.
2. Prove P -> Q.

Example theorem (Using your axioms):

For any integer a, 0.a = 0.

0 + 0 = 0 identity property(+)

(0+0).a = 0.a Multiplication property(=)

0.a + 0.a = 0.a Distributive property

0.a + 0.a + -(0.a) = 0.a + -(0.a) Addition property

0.a + 0 = 0 Inverse property(+)

0.a = 0 Identity property(+)

4)

*Proof by Contradiction* (Indirect proof): (˥ Q -> F)

Goal: q

*Technique*:

1. Assume ˥ q
2. Find a contradiction.
3. ˥ Q -> F it’s contrapositive is T(tautology) -> Q
4. So, T is tautology therefore Q

Example: Euclid’s proof: There must be an infinite number of primes.

5)

Sometimes, Instead of proving statement(Q) by itself, you need to prove combination of them

*Proof by Conjunction*:

Goal: p Λ Q

*Technique*:

1. Prove P
2. Prove Q

Example: Prove that every multiple of 6 is divisible by both 2 and 3.

1. Every multiple of 6 is divisible by 2.
2. Every multiple of 6 is divisible by 3.

6)

*Proof by Disjunction*:

Goal: P V Q

*Technique 1*:

1. Split problem into cases.
2. Prove P in some cases and Q in other cases.

Example: For any integer x, the remainder when is divided by 4 is either 0 or 1.

Case 1: x is even Case 2: x is odd

*Technique 2*:

1. Assume ˥P
2. Prove Q must be true.

Example: For every real number x, if ≥ x then either x ≤ 0 or x ≥ 1.

Assume say x not ≤ 0

X ≥ 1 □

7)

*Goal*: P ↔ Q

Here P ↔ Q is (P → Q) Λ (Q → P)

*Technique*:

1. Prove P → Q is true
2. And then either prove Q → P(Converse) is true or prove ˥P → ˥Q (Inverse) is true.

Problem:

Theorem: For any integer x, x is odd if and only if is odd.

8)

*Proof by Induction*:

Let P(n) be a predicate, If P(0) is true and N, (P(n) → P(n+1)) is true

Then N, P(n) is true.

If P(0), P(0) → P(1), P(1) → P(2), …. are true

Then P(0), P(1), P(2), ….. are true

*Example Theorem*: =

*Pf: By Induction*

Let P(n) be a proposition that =

Base Case: P(0) = = 0

Inductive Step:

Need to prove Show P(n) → P(n+1) is true

Assume P(n) is true for purpose of Induction.

(i.e., assume 1+2+3+…+n =

Need to Show 1+2+3+…+(n+1) =

1+2+3+…+n+(n+1) = =

Definition of leap year

From the beginning of the January 1, 2000 to the end of December 31, 2399 the leap year occurred every 4 years (100 times) minus the 4 years multiple

of 100 (years 2000, 2100, 2200, 2300) but plus 1 year multiple of 400 (year 2000), so its 97 times.

As for multiple of 4, here is how you can approach it. Any number divisible by 100 (that is, with two zeros at the end) is also divisible by 4 because 100=25\*4 and,

therefore, if N=100\*K, then N=(25\*4)\*K=4\*(25\*K) - divisible by 4 (I used associative and commutative properties of multiplication). So, you have to make sure that only

the last two digits of a number make up a number multiple of 4. If this two-digit number xy is combined with any number that has two zeroes at the end like ab00,

you will have a number abxy. Since ab00 is divisible by 4 (since it's divisible by 100) and xy is divisible by 4 (since it's our choice), the combined number

abxy is also divisible by 4. That's how I came up with many numbers divisible by 4 like 2004, 2008 etc.

n4 - (n100 - n400)

Consider one 400 years cycle from 01/01/2000. From the beginning of the January 1, 2000 to the end of December 31, 2399 the leap year occurred every 4 years

(100 times) minus the 4 years multiple of 100 (years 2000, 2100, 2200, 2300) but plus 1 year multiple of 400 (year 2000), so its 97 times. the total number of days

in these 400 years is, therefore, 365\*400+97. The average number of days per year is (365\*400+97)/400 = 365+(97/400) = 365.2425, which is slightly less than 365.25

because of 100 years adjustment and 400 years adjustment.

-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Euclidean division

Given two integers *a* and *b*, with *b* ≠ 0, there exist [unique](http://en.wikipedia.org/wiki/Uniqueness_quantification) integers *q* and *r* such that *a* = *bq* + *r* and 0 ≤ *r* < |*b*|, where |*b*| denotes the [absolute value](http://en.wikipedia.org/wiki/Absolute_value) of *b*.[[1]](http://en.wikipedia.org/wiki/Euclidean_division#cite_note-1)

The four integers that appear in this theorem have been given a name: *a* is called the **dividend**, *b* is called the **divisor**, *q* is called the **quotient** and *r* is called the **remainder**.

The computation of the quotient and the remainder from the dividend and the divisor is called **division** or, in case of ambiguity,**Euclidean division**. The theorem is frequently referred to as the *division algorithm*, although it is a theorem and not an algorithm, because its proof as given below also provides a simple division algorithm for computing *q* and *r*.

Division is not defined in the case where *b* = 0; see [division by zero](http://en.wikipedia.org/wiki/Division_by_zero).

Examples

* If *a* = 7 and *b* = 3, then *q* = 2 and *r* = 1, since 7 = 3 × 2 + 1.
* If *a* = 7 and *b* = −3, then *q* = −2 and *r* = 1, since 7 = −3 × (−2) + 1.
* If *a* = −7 and *b* = 3, then *q* = −3 and *r* = 2, since −7 = 3 × (−3) + 2.
* If *a* = −7 and *b* = −3, then *q* = 3 and *r* = 2, since −7 = −3 × 3 + 2.

This rule will help last 2 cases, first find the largest multiple(q) when multiplied by b will be less than or equal to the number(a) and then what's remaining is the remainder

<http://math.stackexchange.com/questions/19933/does-a-negative-number-really-exist>

**Proof**

The proof consists of two parts — first, the proof of the existence of *q* and *r*, and second, the proof of the uniqueness of *q* and *r*.

### Existence

Consider first the case *b* < 0. Setting *b'* = −*b* and *q'* = −*q*, the equation *a* = *bq* + *r* may be rewritten *a* = *b'q'* + *r* and the inequality 0 ≤ r < |*b*| may be rewritten 0 ≤ r < |*b'*|. This reduces the existence for the case *b* < 0 to that of the case *b* > 0.

Similarly, if *a* < 0 and *b* > 0, setting *a'* = −*a*, *q'* = −*q* − 1 and *r'* = *b* − *r*, the equation *a* = *bq* + *r* may be rewritten *a'* = *bq'* + *r'* and the inequality 0 ≤ *r* < *b* may be rewritten 0 ≤ *r'* < *b*. Thus the proof of the existence is reduced to the case *a* ≥ 0 and *b* > 0 and we consider only this case in the remainder of the proof.

Let *q*1 and *r*1, both nonnegative, such that *a* = *bq*1 + *r*1, for example *q*1 = 0 and *r*1 = *a*. If *r*1 < *b*, we are done. Otherwise *q*2 = *q*1 + 1 and *r*2 = *r*1 − *b* satisfy *a* = *bq*2 + *r*2 and 0 ≤ *r*2 <*r*1. Repeating this process one gets eventually *q* = *qk* and *r* = *rk* such that *a* = *bq* + *r* and 0 ≤ *r* < *b*.

This proves the existence and also gives a simple [division algorithm](http://en.wikipedia.org/wiki/Division_algorithm) to compute the quotient and the remainder. However this algorithm needs *q* steps and is thus not efficient.

### Uniqueness

Suppose there exists *q*, *q'* , *r*, *r'* with 0 ≤ *r*, *r'* < *|b|* such that *a* = *bq* + *r* and *a* = *bq'* + *r'* . Adding the two inequalities 0 ≤ *r* < |*b*| and −|*b*| < −*r'* ≤ 0 yields −|*b*| < *r* − *r'* < |*b*|, that is |*r* − *r'*| < |*b*|.

Subtracting the two equations yields: *b*(*q'*  − *q*) = (*r* − *r'* ). Thus |*b*| divides |*r* − *r'* |. If |*r* − *r'* | ≠ 0 this implies |*b*| < |*r* − *r'* |, contradicting previous inequality. Thus, *r* = *r'* and *b*(*q'*  − *q*) = 0. As *b* ≠ 0, this implies *q* = *q'* , proving uniqueness.

**Program to perform modulo operation:**

There's a rather neat formula for this that works when n < 0 and d > 0: take the bitwise complement of n, do the division, and then take the bitwise complement of the result.

int ifloordiv(int n, int d)

{

if (n >= 0)

return n / d;

else

return ~(~n / d);

}

For the remainder, a similar construction works (compatible with ifloordiv in the sense that the usual invariant ifloordiv(n, d) \* d + ifloormod(n, d) == n is satisfied) giving a result that's always in the range [0, d).

int ifloormod(int n, int d)

{

if (n >= 0)

return n % d;

else

return d + ~(~n % d);

}

For negative divisors, the formulas aren't quite so neat. Here are expanded versions of ifloordiv andifloormod that follow your 'nice-to-have' behavior option (b) for negative divisors.

int ifloordiv(int n, int d)

{

if (d >= 0)

return n >= 0 ? n / d : ~(~n / d);

else

return n <= 0 ? n / d : (n - 1) / d - 1;

}

int ifloormod(int n, int d)

{

if (d >= 0)

return n >= 0 ? n % d : d + ~(~n % d);

else

return n <= 0 ? n % d : d + 1 + (n - 1) % d;

}

For d < 0, there's an unavoidable problem case when d == -1 and n is Integer.MIN\_VALUE, since then the mathematical result overflows the type. In that case, the formula above returns the wrapped result, just as the usual Java division does. As far as I'm aware, this is the only corner case where we silently get 'wrong' results.

Note:

Any negative number n, after 2’s complement changes the sign of n.

Any negative number n, after 1’s complement changes the sign of n and becomes n-1.

-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**GCD**

Here is the recursive method:  
54 = 24\*2+6  
24 = 6\*4 + 0  
Therefore, 6 is the GCD of 54 and 24. This method produces the same result.  
  
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Random -Number Generation

Random Number = Uniform (0, 1)

Random Variate = Other distributions = Function(Random number)

A Sample Generator



For example,





The first 32 numbers obtained by the above procedure

10, 3, 0, 1, 6, 15, 12, 13, 2, 11, 8, 9, 14, 7, 4, 5

10, 3, 0, 1, 6, 15, 12, 13, 2, 11, 8, 9, 14, 7, 4, 5.

By dividing x's by 16:

0.6250, 0.1875, 0.0000, 0.0625, 0.3750, 0.9375, 0.7500,

0.8125, 0.1250, 0.6875, 0.5000, 0.5625, 0.8750, 0.4375,

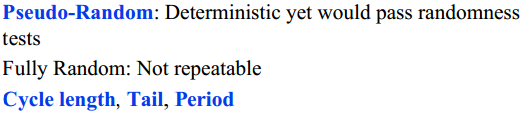
0.2500, 0.3125, 0.6250, 0.1875, 0.0000, 0.0625, 0.3750,

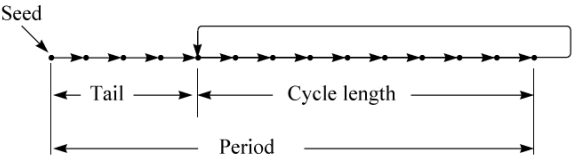
0.9375, 0.7500, 0.8125, 0.1250, 0.6875, 0.5000, 0.5625,

0.8750, 0.4375, 0.2500, 0.3125.

Terminology





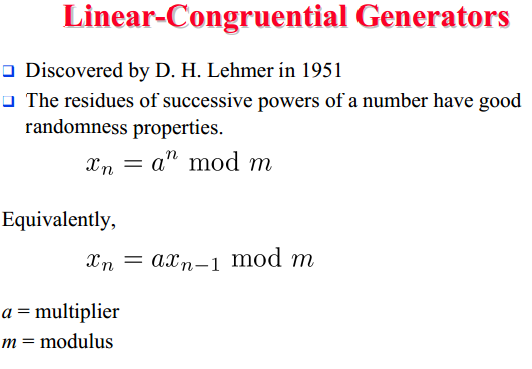


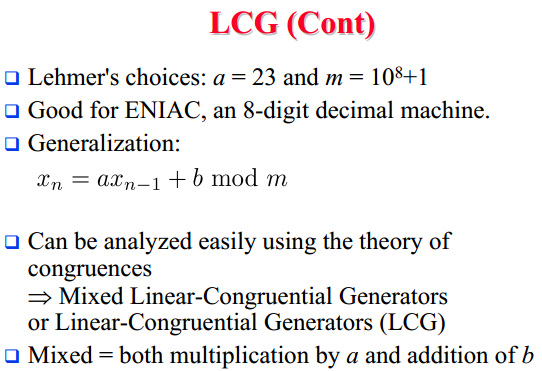
Properties of a Good Generator

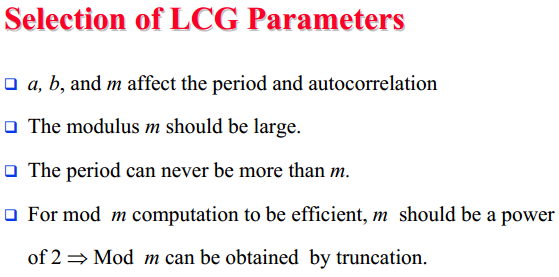
* It should be efficiently computable.
* The period should be large.
* The successive values should be independent and uniformly distributed

Types of Generators

* + Linear congruential generators
  + Tausworthe generators
  + Extended Fibonacci generators
  + Combined generators







--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Newton Raphson method.**

This link will exhibit you, How slope concept will help us reach the point where f(x) is zero.

<http://en.wikipedia.org/wiki/Newton's_method#mediaviewer/File:NewtonIteration_Ani.gif>

We have to come up with an iterative process of finding a point x where a function f(x)=x^2-7 equals to zero.

We start from any x1 and, using the formula x2 = x1 - f(x1)/f'(x1), come up with x2 that is closer to a solution.

Then go to x3 = x2 - f(x2)/f'(x2), then x4 = x3 - f(x3)/f'(x3) etc.

Under relatively weak conditions this process produces points x1, x2, x3.... that come closer and closer to a solution. Graphically it seems obvious.

In your example:

f(x)=x^2-7 and

f'(x)=2x

Let's start with x1=3 (so, x1^2=9 not perfect)  
Then x2 = 3 - (3^2-7)/(2\*3) = 8/3 (now x2^2=7.111111, which is better)  
x3 = (8/3) - (64/9 - 7)/(16/3) = 127/48 (x3^2=7.000434, even better)

etc.

Idea is, If I need to find square root of 7, then consider a function *f(x) = x^2 – 7* where *x:f(x)=0*.

Using Newton Raphson, find that *x* by considering some initial point(guess) and concept(or God’s magic) of slope at each better point is making you reach that x

*Formulae for finding better x2 than previous x1 can be found as shown*

Let's use some letters to mark the important points.

The origin of coordinates - point O

Point where a tangent touches the function - point A

Point where a tangent intersects the X-axis - point B

Point of intersection of the perpendicular dropped from point A down and the X-axis point C.

OB=x2

OC=x1

AC=f(x1)

BC=x1-x2

In a triangle ABC: AC/BC = tangent of an angle between AB and X-axis, which is a derivative of f(x) at point x1.

Now, BC\*tan(angle between AB and X-axis)=AC

Therefore, BC\*f'(x1)=f(x1)

Hence, (x1-x2)\*f'(x1)=f(x1)

and x1-x2 = f(x1) / f'(x1)

x2 = x1 - f(x1)/f'(x1)

-----------------------------------------------------------------------------------------------------------------------------------

Prime number:

A **composite number** is a positive integer that has at least one positive divisor other than one or the number itself.

Sieve of Eratosthenes:

All integers are assumed prime until proven composite.