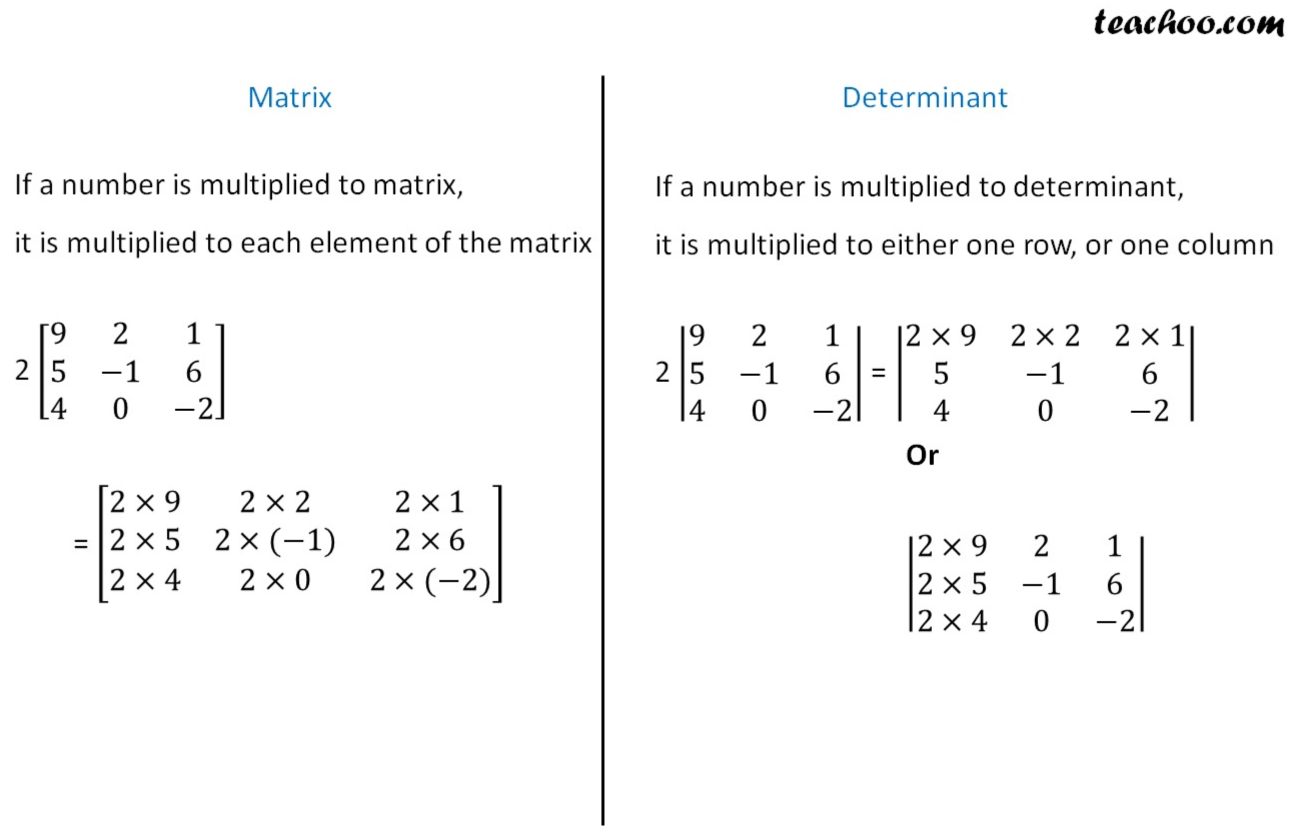
PROGRAMMING MATH & FUNCTIONS

1)Matrix (матрица, детерминант(определитель))



Matrix

If a number is multiplied to matrix,

it is multiplied to each element of the matrix

$$2\begin{bmatrix}9 & 2 & 1\\ 5 & -1 & 6\\ 4 & 0 & -2\end{bmatrix}$$

$$=\begin{bmatrix}2\times 9 & 2\times 2 & 2\times 1\\ 2\times 5 & 2\times(-1) & 2\times 6\\ 2\times 4 & 2\times 0 & 2\times(-2)\end{bmatrix}$$

Determinant

If a number is multiplied to determinant,

it is multiplied to either one row, or one column

$$2\begin{vmatrix}9 & 2 & 1\\ 5 & -1 & 6\\ 4 & 0 & -2\end{vmatrix}=\begin{vmatrix}2\times 9 & 2\times 2 & 2\times 1\\ 5 & -1 & 6\\ 4 & 0 & -2\end{vmatrix}$$

Or

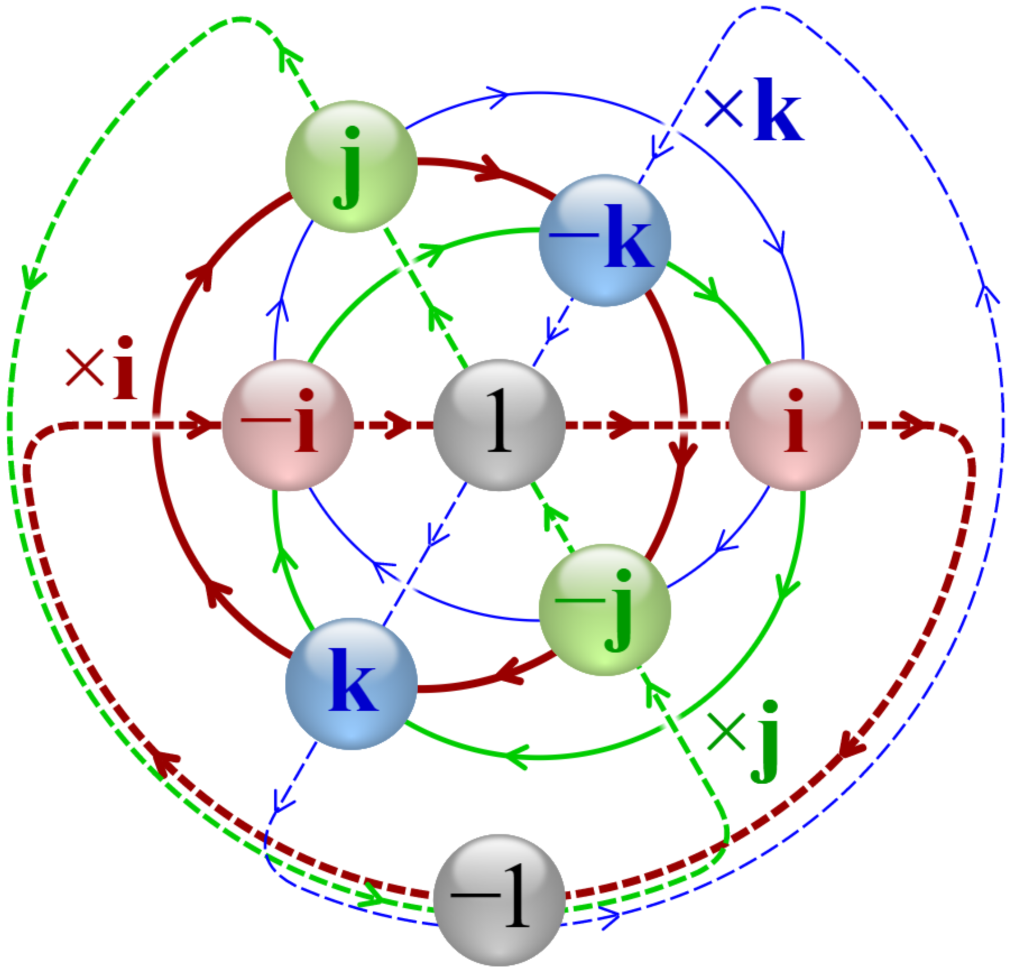
$$\begin{vmatrix}2\times 9 & 2 & 1\\ 2\times 5 & -1 & 6\\ 2\times 4 & 0 & -2\end{vmatrix}$$

3)Quaternion

In mathematics, the **quaternion** number system extends the complex numbers. Quaternions were first described by Irish mathematician William Rowan Hamilton in 1843[1][2] and applied to mechanics in three-dimensional space. Hamilton defined a quaternion as the quotient of two *directed lines* in a three-dimensional space,[3] or, equivalently, as the quotient of two vectors.[4] Multiplication of quaternions is noncommutative.

Quaternion multiplication table

	1	**i**	**j**	**k**
1	1	i	j	k
i	i	−1	k	−j
j	j	−k	−1	i
k	k	j	−i	−1

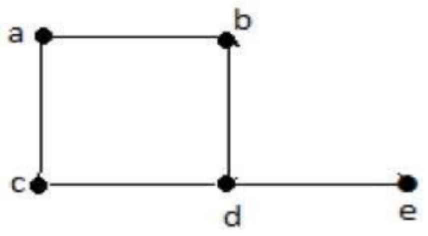


4) Graph Theory (More here: https://www.tutorialspoint.com/graph_theory/graph_theory_quick_guide.htm)

What is a Graph?

A graph is a pictorial representation of a set of objects where some pairs of objects are connected by links. The interconnected objects are represented by points termed as **vertices**, and the links that connect the vertices are called **edges**.

Formally, a graph is a pair of sets **(V, E)**, where **V** is the set of vertices and E is the set of edges, connecting the pairs of vertices. Take a look at the following graph −



In the above graph,

V = {a, b, c, d, e}

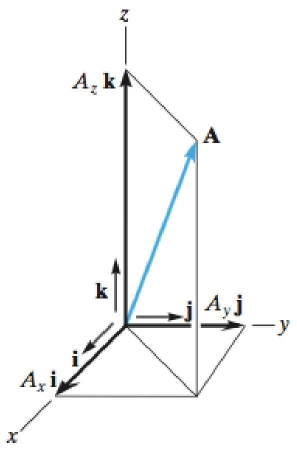
E = {ab, ac, bd, cd, de}

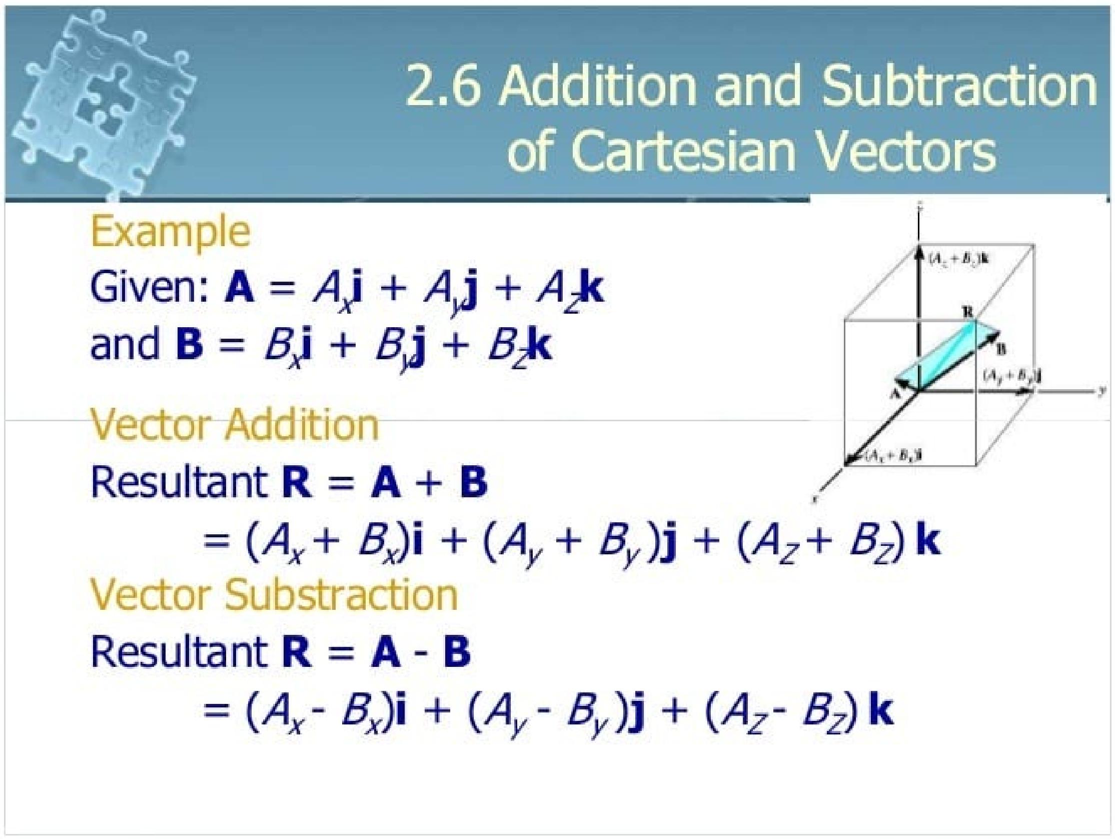
5) Cartesian Vector

Cartesian Vector Representation. Since the three components of **A** in Eq. 2–2 act in the positive **i**, **j**, and **k** directions, Fig. 2–24, we can write **A** in Cartesian vector form as

$$\boxed{\mathbf{A} = A_x\mathbf{i} + A_y\mathbf{j} + A_z\mathbf{k}} \qquad (2\text{–}3)$$

There is a distinct advantage to writing vectors in this manner. Separating the *magnitude* and *direction* of each *component vector* will simplify the operations of vector algebra, particularly in three dimensions.



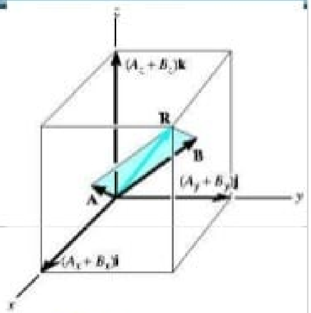


2.6 Addition and Subtraction of Cartesian Vectors

Example

Given: $\mathbf{A} = A_x\mathbf{i} + A_y\mathbf{j} + A_Z\mathbf{k}$

and $\mathbf{B} = B_x\mathbf{i} + B_y\mathbf{j} + B_Z\mathbf{k}$



Vector Addition

Resultant $\mathbf{R} = \mathbf{A} + \mathbf{B}$

$= (A_x + B_x)\mathbf{i} + (A_y + B_y)\mathbf{j} + (A_Z + B_Z)\mathbf{k}$

Vector Substraction

Resultant $\mathbf{R} = \mathbf{A} - \mathbf{B}$

$= (A_x - B_x)\mathbf{i} + (A_y - B_y)\mathbf{j} + (A_Z - B_Z)\mathbf{k}$

6) Vectors

Basic Formulas

Components

$$\vec{AB} = (x_2 - x_1, y_2 - y_1, z_2 - z_1)$$

Magnitude or Length

$$\vec{u} = (u_1, u_2, u_3)$$

$$|\vec{u}| = \sqrt{u_1^2 + u_2^2 + u_3^2}$$

Distance between two points

$$d(A, B) = \left|\vec{AB}\right| = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

Unit Vector

$$\vec{u}_{\vec{v}} = \frac{\vec{v}}{|\vec{v}|}$$

Vector Addition

$$\vec{u} = (u_1, u_2, u_3) \qquad \vec{v} = (v_1, v_2, v_3)$$

$$\vec{u} + \vec{v} = (u_1 + v_1, u_2 + v_2, u_3 + v_3)$$

Scalar Multiplication

$$k.\vec{u} = (ku_1 + ku_2 + ku_3)$$

Linearly Dependent Vectors

$$\vec{u} = k\vec{v} \qquad (u_1, u_2, u_3) = (kv_1 + kv_2 + kv_3)$$

$$\frac{u_1}{v_1} = \frac{u_2}{v_2} = \frac{u_3}{v_3} = k$$

$$\begin{vmatrix} u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \\ w_1 & w_2 & w_3 \end{vmatrix} = 0$$

Linearly Independent Vectors

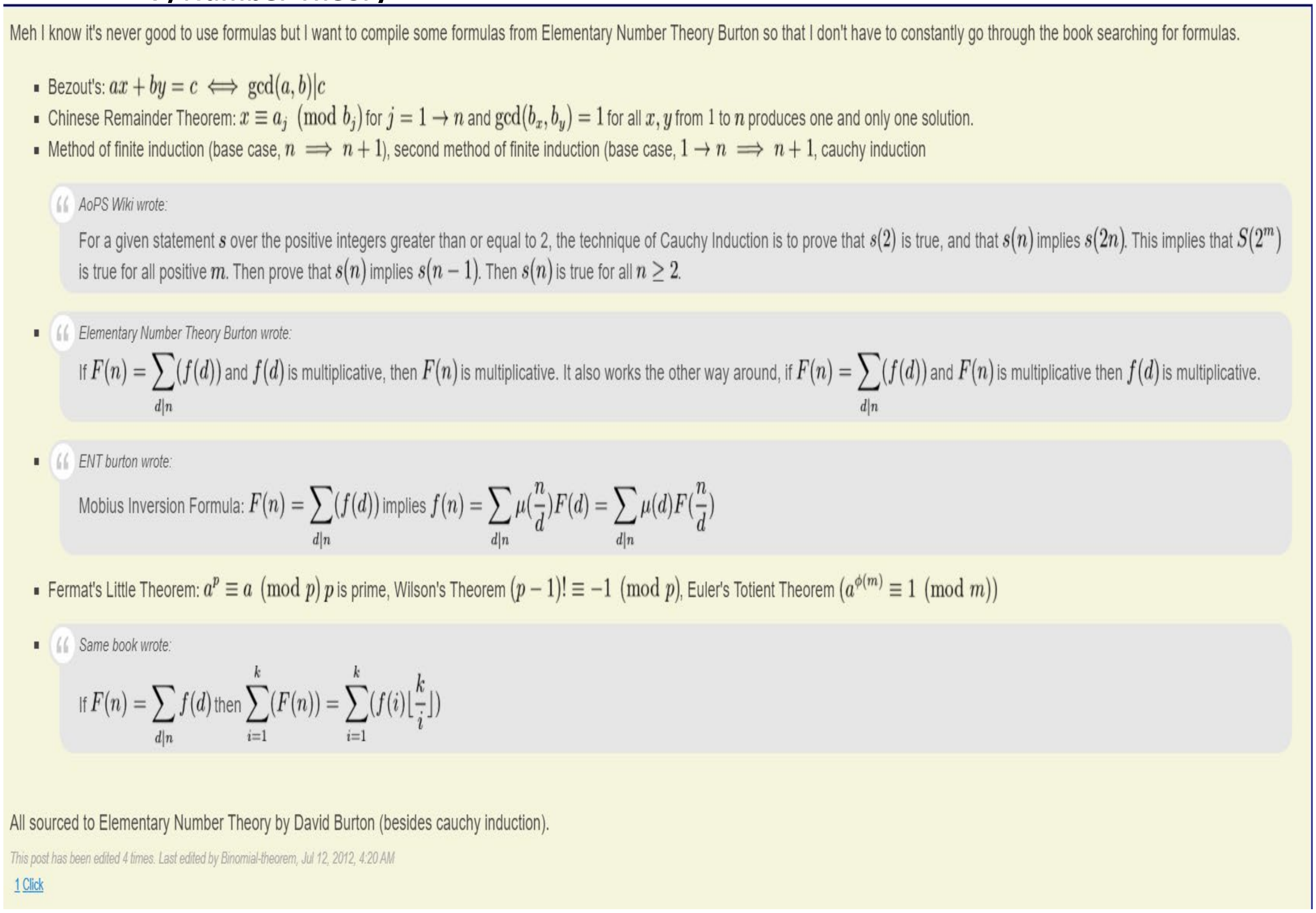
$$\begin{vmatrix} u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \\ w_1 & w_2 & w_3 \end{vmatrix} \neq 0$$

Dot Product

$$\vec{u}.\vec{v} = |\vec{u}| \,.\, |\vec{v}| \,.\cos\alpha$$

$$\vec{u}.\vec{v} = u_1.v_1 + u_2.v_2 + u_3.v_3$$

7) Number Theory



Meh I know it's never good to use formulas but I want to compile some formulas from Elementary Number Theory Burton so that I don't have to constantly go through the book searching for formulas.

- Bezout's: $ax + by = c \iff \gcd(a, b)|c$
- Chinese Remainder Theorem: $x \equiv a_j \pmod{b_j}$ for $j = 1 \to n$ and $\gcd(b_x, b_y) = 1$ for all x, y from 1 to n produces one and only one solution.
- Method of finite induction (base case, $n \implies n+1$), second method of finite induction (base case, $1 \to n \implies n+1$, cauchy induction

  AoPS Wiki wrote:

  For a given statement s over the positive integers greater than or equal to 2, the technique of Cauchy Induction is to prove that $s(2)$ is true, and that $s(n)$ implies $s(2n)$. This implies that $S(2^m)$ is true for all positive m. Then prove that $s(n)$ implies $s(n-1)$. Then $s(n)$ is true for all $n \geq 2$.

- Elementary Number Theory Burton wrote:

  If $F(n) = \sum_{d|n}(f(d))$ and $f(d)$ is multiplicative, then $F(n)$ is multiplicative. It also works the other way around, if $F(n) = \sum_{d|n}(f(d))$ and $F(n)$ is multiplicative then $f(d)$ is multiplicative.

- ENT burton wrote:

  Mobius Inversion Formula: $F(n) = \sum_{d|n}(f(d))$ implies $f(n) = \sum_{d|n}\mu(\frac{n}{d})F(d) = \sum_{d|n}\mu(d)F(\frac{n}{d})$

- Fermat's Little Theorem: $a^p \equiv a \pmod{p}$ p is prime, Wilson's Theorem $(p-1)! \equiv -1 \pmod{p}$, Euler's Totient Theorem $(a^{\phi(m)} \equiv 1 \pmod{m})$

- Same book wrote:

  If $F(n) = \sum_{d|n} f(d)$ then $\sum_{i=1}^{k}(F(n)) = \sum_{i=1}^{k}(f(i)\lfloor\frac{k}{i}\rfloor)$

All sourced to Elementary Number Theory by David Burton (besides cauchy induction).

This post has been edited 4 times. Last edited by Binomial-theorem, Jul 12, 2012, 4:20 AM

1 Click

8)Calculus theory

Calculus is the branch of mathematics, which deals in the study rate of change and its application in solving the equations. Differential Calculus deals with the rates of change and slopes of curves. Integral Calculus deals mainly with the accumulation of quantities and the areas under and between curves. In calculus, we use the fundamental notions of convergence of infinite sequences and infinite series to a well-defined limit. its value is less than any positive real number. According to this point of view, we can say that calculus is a collection of techniques for manipulating infinitesimals. In calculus the symbols dxdxanddydy were taken to be infinitesimal, and the derivative dydxdy/dx was simply their ratio.

Differential Calculus

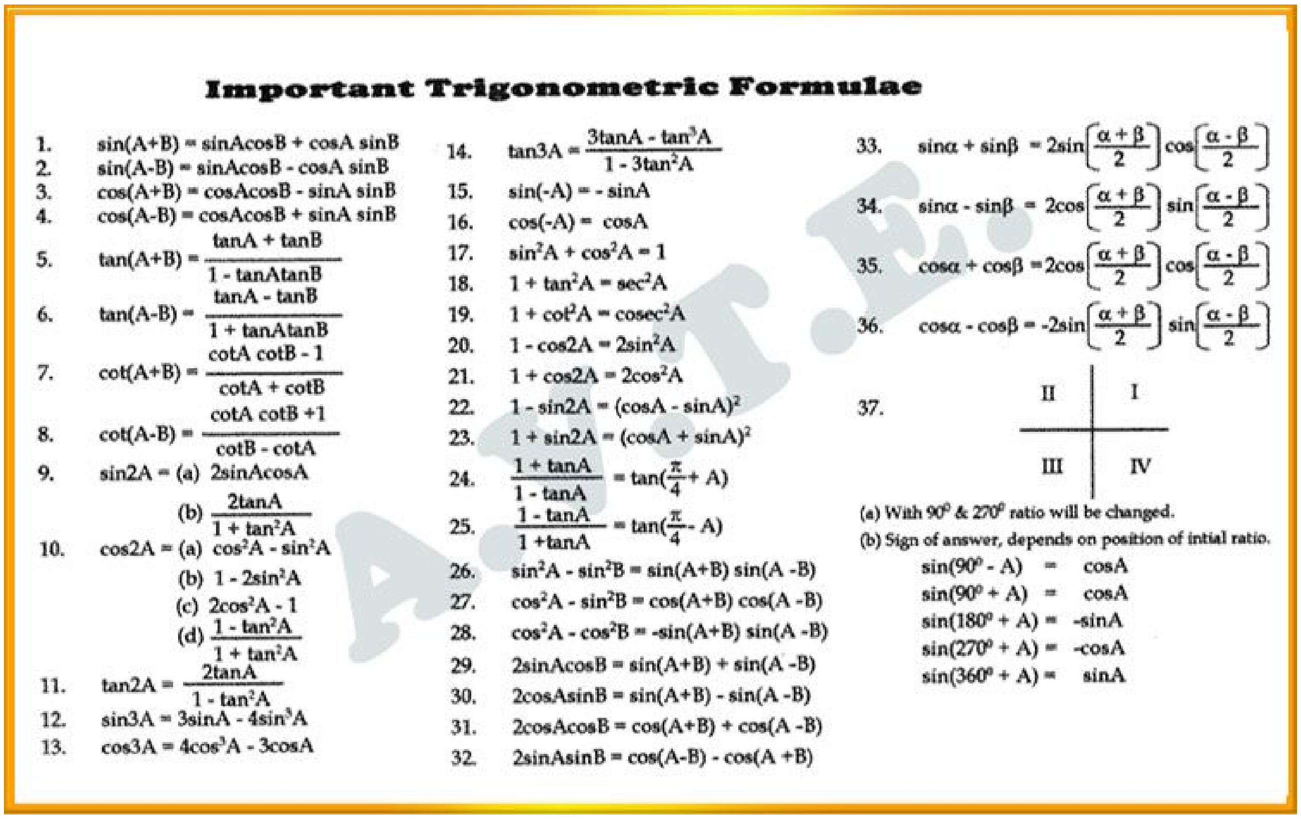
- $\frac{\mathrm{d}r^2}{\mathrm{d}x} = nx^{(}n-1)$
- $\frac{\mathrm{d}(fg)}{\mathrm{d}x} = fg^1 + gf^1$
- $\frac{\mathrm{d}}{\mathrm{d}x}\left(\frac{f}{g}\right)= \frac{gf^1-fg^1}{g^2}$
- $\frac{\mathrm{d}f(g(x))}{\mathrm{d}x} = f^1(g(x))g^1(x)$
- $\frac{\mathrm{d}(sinx)}{\mathrm{d}x} = cosx$
- $\frac{\mathrm{d}(cosx)}{\mathrm{d}x} = -sinx$
- $\frac{\mathrm{d}(tanx)}{\mathrm{d}x} = -sec2x$
- $\frac{\mathrm{d}(cotx)}{\mathrm{d}x} = csc2x$
- $\frac{\mathrm{d}(secx)}{\mathrm{d}x} = secxtanx$
- $\frac{\mathrm{d}(cscx)}{\mathrm{d}x} = -cscxcotx$
- $\frac{\mathrm{d}(e^x)}{\mathrm{d}x} = e^x$
- $\frac{\mathrm{d}(a^x)}{\mathrm{d}x} = a^x \ln a$
- $\frac{\mathrm{d}}{\mathrm{d}x}\ln x = \frac{1}{x}$
- $\frac{\mathrm{d}(arcsinx)}{\mathrm{d}x} = \frac{1}{\sqrt{1-x^2}}$
- $\frac{\mathrm{d}(arcsinx)}{\mathrm{d}x} = \frac{1}{\sqrt{1+x^2}}$

Integral Calculus

- $\int adr = ax + C$
- $\int \frac{1}{x}dr = \ln|x| + C$
- $\int axdx = \frac{ex}{\ln a} + C$
- $\int \ln xdx = x\ln x - x + C$
- $\int sinxdx = -cosx + C$
- $\int cosxdx = sinx + C$
- $\int tanxdr + \ln|secx| + C = -\ln|cosx| + C$
- $\int cotxdr = \ln|sinx| + C$
- $\int secxdx = \ln|secx + tanx| + C$
- $\int cscxdx = \ln|cscx- cotx| + C$
- $\int sec^2xdx = tanx + C$
- $\int secxtanxdx = secx + C$
- $\int csc^2xdr = -cotx + C$
- $\int tan^2xdr = tanx - x + C$
- $\int fracdr\sqrt{a^2 - x^2} = arcsin\frac{x}{a} + C$
- $\int fracdr\sqrt{a^2 + x^2} = \frac{1}{a}arcsin\frac{x}{a} + C$

9)Trigonometry Formulas

Trigonometry (from Greek *trigōnon*, "triangle" and *metron*, "measure"[1]) is a branch of mathematics that studies relationships between side lengths and angles of triangles



Important Trigonometric Formulae

1. $\sin(A+B) = \sin A\cos B + \cos A \sin B$
2. $\sin(A-B) = \sin A\cos B - \cos A \sin B$
3. $\cos(A+B) = \cos A\cos B - \sin A \sin B$
4. $\cos(A-B) = \cos A\cos B + \sin A \sin B$
5. $\tan(A+B) = \dfrac{\tan A + \tan B}{1 - \tan A\tan B}$
6. $\tan(A-B) = \dfrac{\tan A - \tan B}{1 + \tan A\tan B}$
7. $\cot(A+B) = \dfrac{\cot A \cot B - 1}{\cot A + \cot B}$
8. $\cot(A-B) = \dfrac{\cot A \cot B + 1}{\cot B - \cot A}$
9. $\sin 2A =$ (a) $2\sin A\cos A$
   (b) $\dfrac{2\tan A}{1 + \tan^2 A}$
10. $\cos 2A =$ (a) $\cos^2 A - \sin^2 A$
    (b) $1 - 2\sin^2 A$
    (c) $2\cos^2 A - 1$
    (d) $\dfrac{1 - \tan^2 A}{1 + \tan^2 A}$
11. $\tan 2A = \dfrac{2\tan A}{1 - \tan^2 A}$
12. $\sin 3A = 3\sin A - 4\sin^3 A$
13. $\cos 3A = 4\cos^3 A - 3\cos A$
14. $\tan 3A = \dfrac{3\tan A - \tan^3 A}{1 - 3\tan^2 A}$
15. $\sin(-A) = -\sin A$
16. $\cos(-A) = \cos A$
17. $\sin^2 A + \cos^2 A = 1$
18. $1 + \tan^2 A = \sec^2 A$
19. $1 + \cot^2 A = \mathrm{cosec}^2 A$
20. $1 - \cos 2A = 2\sin^2 A$
21. $1 + \cos 2A = 2\cos^2 A$
22. $1 - \sin 2A = (\cos A - \sin A)^2$
23. $1 + \sin 2A = (\cos A + \sin A)^2$
24. $\dfrac{1 + \tan A}{1 - \tan A} = \tan(\frac{\pi}{4} + A)$
25. $\dfrac{1 - \tan A}{1 + \tan A} = \tan(\frac{\pi}{4} - A)$
26. $\sin^2 A - \sin^2 B = \sin(A+B)\sin(A-B)$
27. $\cos^2 A - \sin^2 B = \cos(A+B)\cos(A-B)$
28. $\cos^2 A - \cos^2 B = -\sin(A+B)\sin(A-B)$
29. $2\sin A\cos B = \sin(A+B) + \sin(A-B)$
30. $2\cos A\sin B = \sin(A+B) - \sin(A-B)$
31. $2\cos A\cos B = \cos(A+B) + \cos(A-B)$
32. $2\sin A\sin B = \cos(A-B) - \cos(A+B)$
33. $\sin\alpha + \sin\beta = 2\sin\left(\dfrac{\alpha+\beta}{2}\right)\cos\left(\dfrac{\alpha-\beta}{2}\right)$
34. $\sin\alpha - \sin\beta = 2\cos\left(\dfrac{\alpha+\beta}{2}\right)\sin\left(\dfrac{\alpha-\beta}{2}\right)$
35. $\cos\alpha + \cos\beta = 2\cos\left(\dfrac{\alpha+\beta}{2}\right)\cos\left(\dfrac{\alpha-\beta}{2}\right)$
36. $\cos\alpha - \cos\beta = -2\sin\left(\dfrac{\alpha+\beta}{2}\right)\sin\left(\dfrac{\alpha-\beta}{2}\right)$
37. $$\begin{array}{c|c} \text{II} & \text{I} \\ \hline \text{III} & \text{IV} \end{array}$$

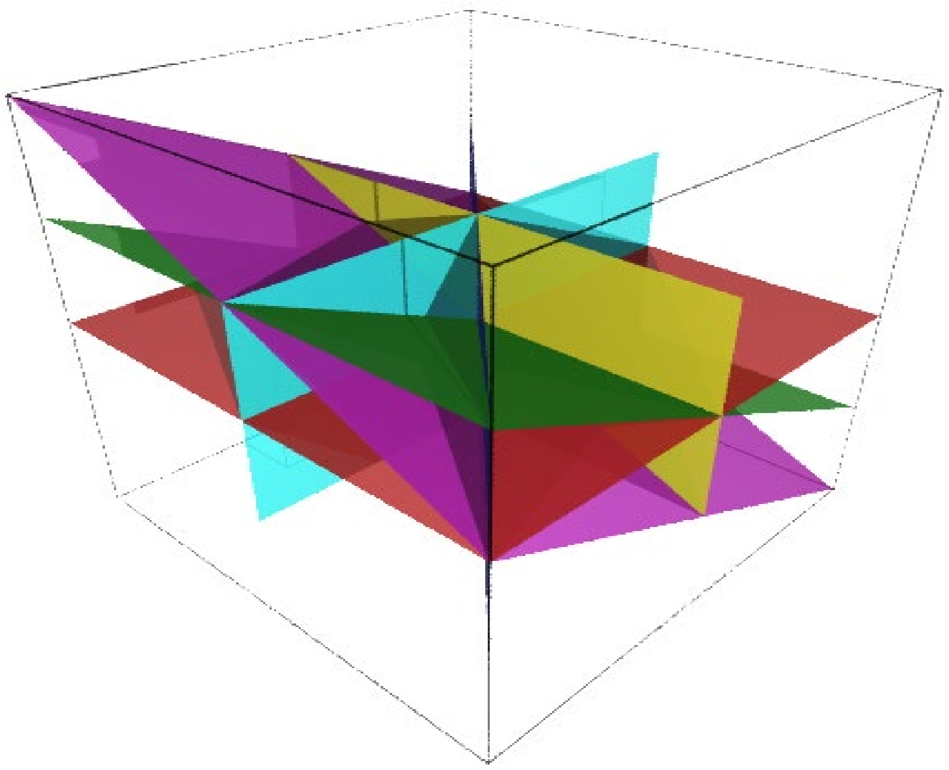
(a) With 90° & 270° ratio will be changed.
(b) Sign of answer, depends on position of intial ratio.

$\sin(90° - A) = \cos A$
$\sin(90° + A) = \cos A$
$\sin(180° + A) = -\sin A$
$\sin(270° + A) = -\cos A$
$\sin(360° + A) = \sin A$

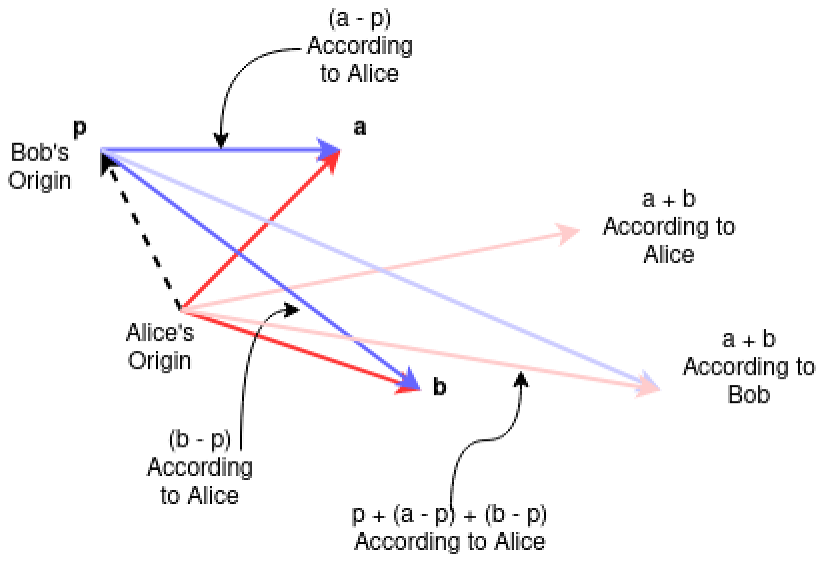
10)Geometry

In geometry, a **hyperplane** is a subspace whose dimension is one less than that of its ambient space. If a space is 3-dimensional then its hyperplanes are the 2-dimensional planes, while if the space is 2-dimensional, its hyperplanes are the 1-dimensional lines. This notion can be used in any general space in which the concept of the dimension of a subspace is defined.

• A hyperplane is a higher-dimensional generalization of lines and planes. The equation of a hyperplane is w · x + b = 0, where w is a vector normal to the hyperplane and b is an offset. Note that we can multiply by any constant and preserve the equality; if we multiply by 1/kwk, we get a new equation ˆw · x + b 0 = 0, where ˆw = w/kwk is the unit normal vector and b 0 = b/kwk is the distance from the hyperplane to the origin.



In an **affine space**, there is no distinguished point that serves as an origin. Hence, no vector has a fixed origin and no vector can be uniquely associated to a point. In an affine space, there are instead *displacement vectors*, also called *translation* vectors or simply *translations*, between two points of the space.[1] Thus it makes sense to subtract two points of the space, giving a translation vector, but it does not make sense to add two points of the space. Likewise, it makes sense to add a displacement vector to a point of an affine space, resulting in a new point translated from the starting point by that vector.


(a - p) According to Alice
p
Bob's Origin
a
a + b According to Alice
Alice's Origin
a + b According to Bob
b
(b - p) According to Alice
p + (a - p) + (b - p) According to Alice

More: https://en.wikipedia.org/wiki/Affine_space

11) Logic Arithmetic

Uses logical operators in programming language, for every language they different (&&,||,==)

12) Boolean Arithmetic

(True or false)

13) Bit Manipulation

(And, Or, Xor, Not, Xnor, Neg,Nand, Nor, Left Shift, Right Shift)

Bit manipulation is the **act of algorithmically manipulating bits or other pieces of data shorter than a word**. Computer programming tasks that require bit manipulation include low-level device control, error detection and correction algorithms, data compression, encryption algorithms, and optimization.

More on: https://www.hackerearth.com/practice/basic-programming/bit-manipulation/basics-of-bit-manipulation/tutorial/

14)Comparing Numbers

15)Signedness

In computing, signedness is **a property of data types representing numbers in computer programs**. A numeric variable is signed if it can represent both positive and negative numbers, and unsigned if it can only represent non-negative numbers (zero or positive numbers).

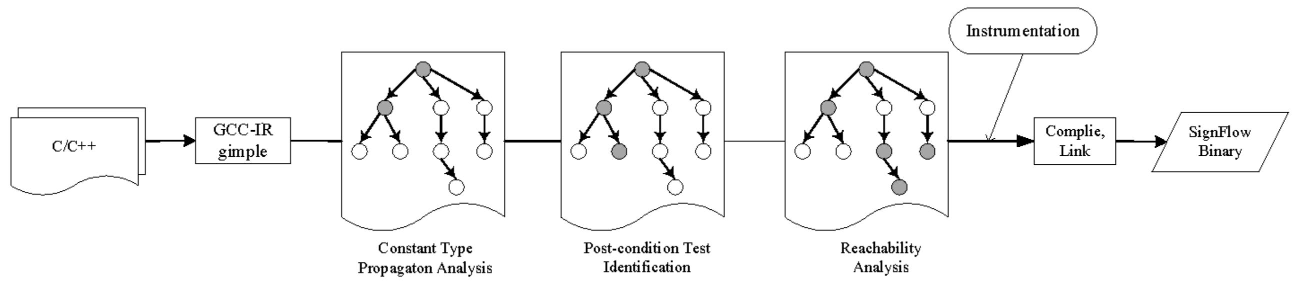


TABLE I. POST-CONDITION TEST FOR SIGNEDNESS CONVERSIONS

Signedness Conversion	Post-condition Test
int x; unsigned int y; x = (int) y;	$x < 0$
unsigned int x; int y; x = (unsigned int) y;	$x > 0x7fffffff$

TABLE II. SANITIZATION OPERATOR FOR SIGNEDNESS ERRORS

Op Type	Basic Form	Influence
bitwise-and	res = a & b, and $b < 0x80000000$	Erased
modulo	res = a % b, and $b < 0x80000000$	Erased
left-shift	res = a $\ll$ b	Replaced
right-shift	res = a $\gg$ b	Replaced
bitwise-xor	res = a $\oplus$ b	Erased
bitwise-not	res = $\sim$ a	Erased

TABLE III. 7 HARMFUL INTEGER SIGNEDNESS BUGS IN REAL WORLD

CVE	Programs	Version	Sign Conv.	Sink
2008-1803	Rdesktop	1.5.0	u→s	bound check
2009-3743	GhostScript	8.70	s→u	memmove
2011-1471	PHP	5.3.6*	s→u	bound check
2012-3368	Dtach	0.8	s→u	bound check
2013-4927	Wireshark	1.10.0	u→s	loop
2013-6489	Pidgin	2.10.11*	s→u	malloc
2014-0045	Mumble	1.2.4	s→u	malloc

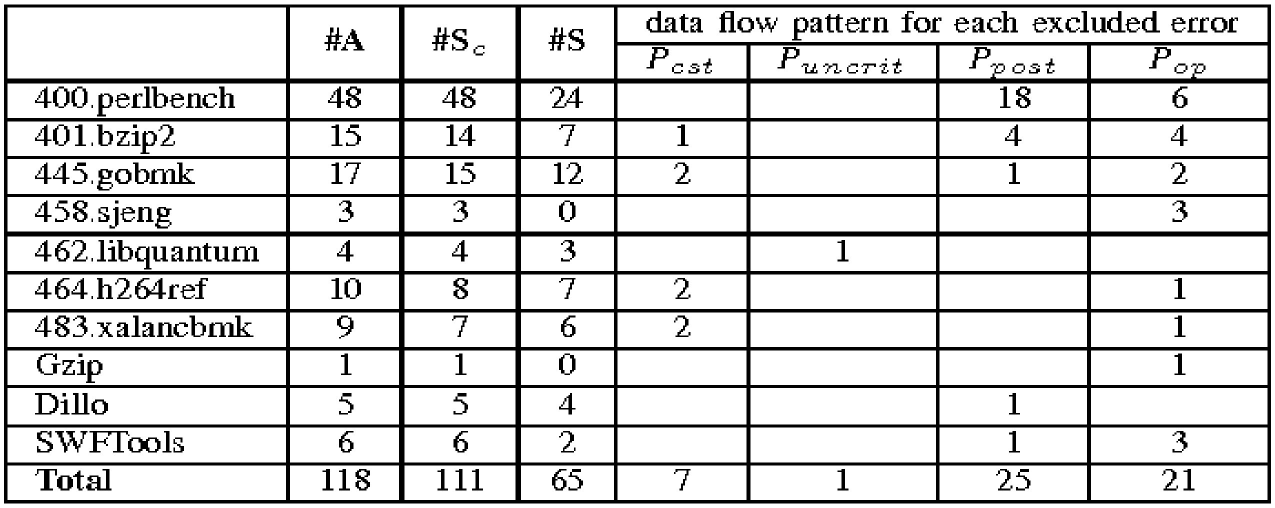
TABLE IV. INTEGER SIGNEDNESS ERRORS REPORTED BY ALL, SIGNFLOW$_{cst}$ AND SIGNFLOW

	#A	#S$_c$	#S	data flow pattern for each excluded error			
				P_{cst}	P_{uncrit}	P_{post}	P_{op}
400.perlbench	48	48	24			18	6
401.bzip2	15	14	7	1		4	4
445.gobmk	17	15	12	2		1	2
458.sjeng	3	3	0				3
462.libquantum	4	4	3		1		
464.h264ref	10	8	7	2			1
483.xalancbmk	9	7	6	2			1
Gzip	1	1	0				1
Dillo	5	5	4			1	
SWFTools	6	6	2			1	3
Total	118	111	65	7	1	25	21

TABLE III. 7 HARMFUL INTEGER SIGNEDNESS BUGS IN REAL WORLD

CVE	Programs	Version	Sign Conv.	Sink
2008-1803	Rdesktop	1.5.0	u→s	bound check
2009-3743	GhostScript	8.70	s→u	memmove
2011-1471	PHP	5.3.6*	s→u	bound check
2012-3368	Dtach	0.8	s→u	bound check
2013-4927	Wireshark	1.10.0	u→s	loop
2013-6489	Pidgin	2.10.11*	s→u	malloc
2014-0045	Mumble	1.2.4	s→u	malloc

TABLE IV. INTEGER SIGNEDNESS ERRORS REPORTED BY ALL, SIGNFLOW$_{cst}$ AND SIGNFLOW



	#A	#S$_c$	#S	data flow pattern for each excluded error			
				P_{cst}	P_{uncrit}	P_{post}	P_{op}
400.perlbench	48	48	24			18	6
401.bzip2	15	14	7	1		4	4
445.gobmk	17	15	12	2		1	2
458.sjeng	3	3	0				3
462.libquantum	4	4	3		1		
464.h264ref	10	8	7	2			1
483.xalancbmk	9	7	6	2			1
Gzip	1	1	0				1
Dillo	5	5	4			1	
SWFTools	6	6	2			1	3
Total	118	111	65	7	1	25	21

16) Add, Subtract, Multiply, Division

17) Tables

LogTables

Every SQL Server database has a **transaction log that records all transactions and the database modifications made by each transaction**. The transaction log is a critical component of the database. If there is a system failure, you will need that log to bring your database back to a consistent state

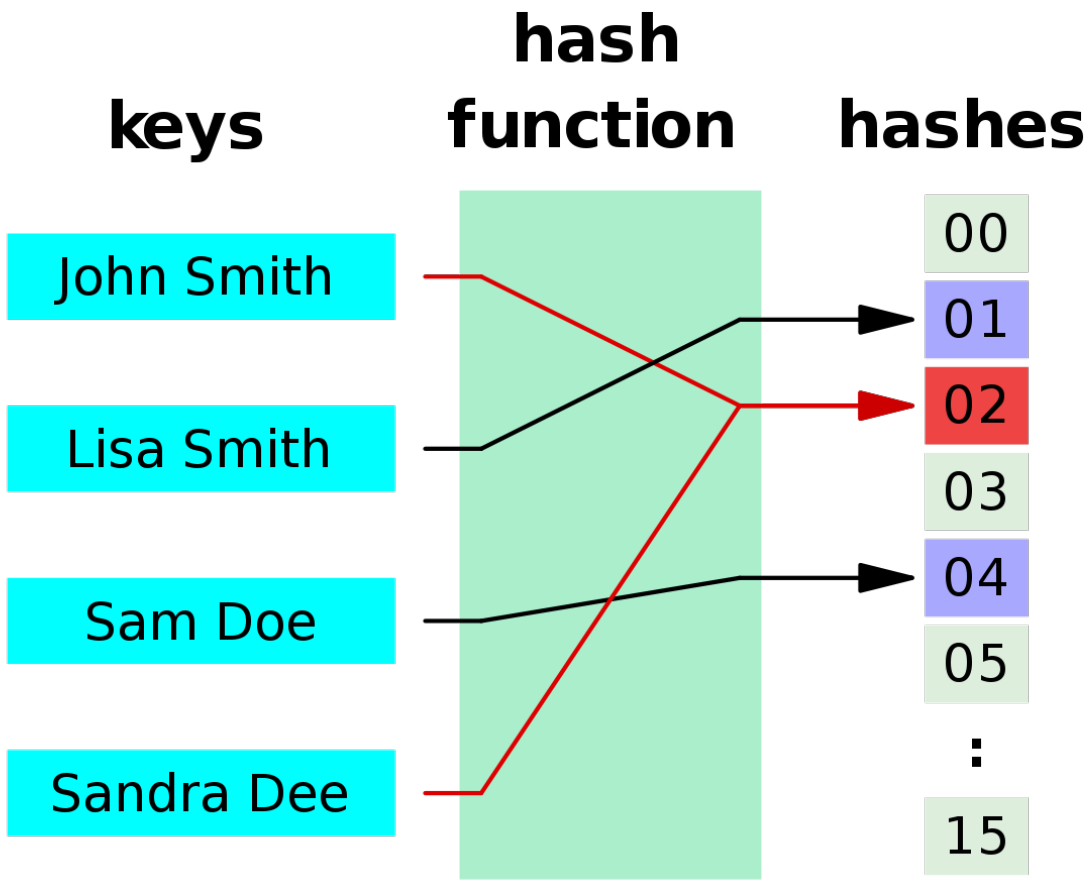
Table logging is **a track mechanism**, that makes possible to record user activity in a system It is useful if you want to know: Who made a change? ... As the parameter rec/client is used for controlling table logging at R/3 system level

Division Tables

Businesses often require reports that require more than the classic set operators. Surprisingly, a business requirement can often be expressed neatly in terms of the DIVISION relationship operator: How can this be done with SQL Server? Joe Celko opens up the 'Manga Guide to Databases', meets the Database Fairy, and is inspired to explain DIVISION.

18) Hash Function

A **hash function** is any function that can be used to map data of arbitrary size to fixed-size values. The values returned by a hash function are called *hash values*, *hash codes*, *digests*, or simply *hashes*. The values are usually used to index a fixed-size table called a *hash table*. Use of a hash function to index a hash table is called *hashing* or *scatter storage addressing*.



20) Modulo (%)

In computing, the modulo operation **returns the remainder or signed remainder of a division**, after one number is divided by another (called the modulus of the operation). ... The modulo operation is to be distinguished from the symbol mod, which refers to the modulus (or divisor) one is operating from.

21) Power/Exponent(pow)

Return Type: The function returns the number base raised to the power. The type of this method is *System.Double*

Examples:

Input : base = 8, power =2

Output : 64

Input : base = 2.5, power =3

Output : 15.625

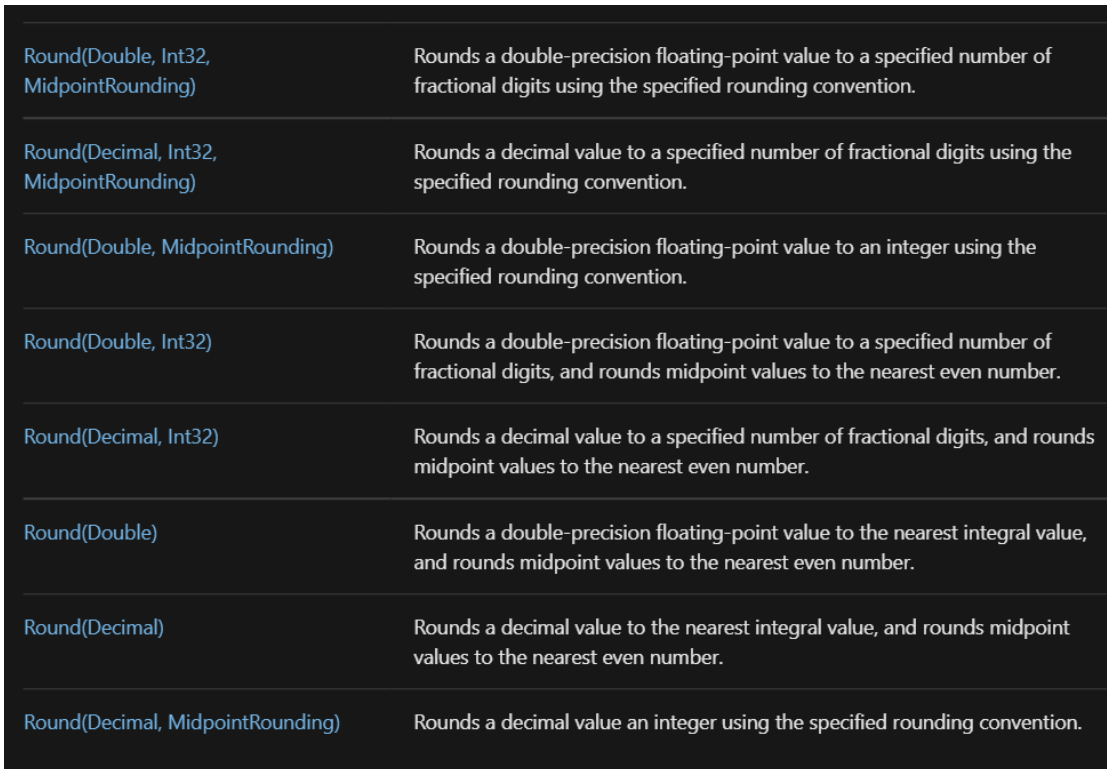
22)Absolute numbers (abs)

The `Math.abs()` function returns the absolute value of a number. That is, it returns `x` if `x` is positive or zero, and the negation of `x` if `x` is negative.

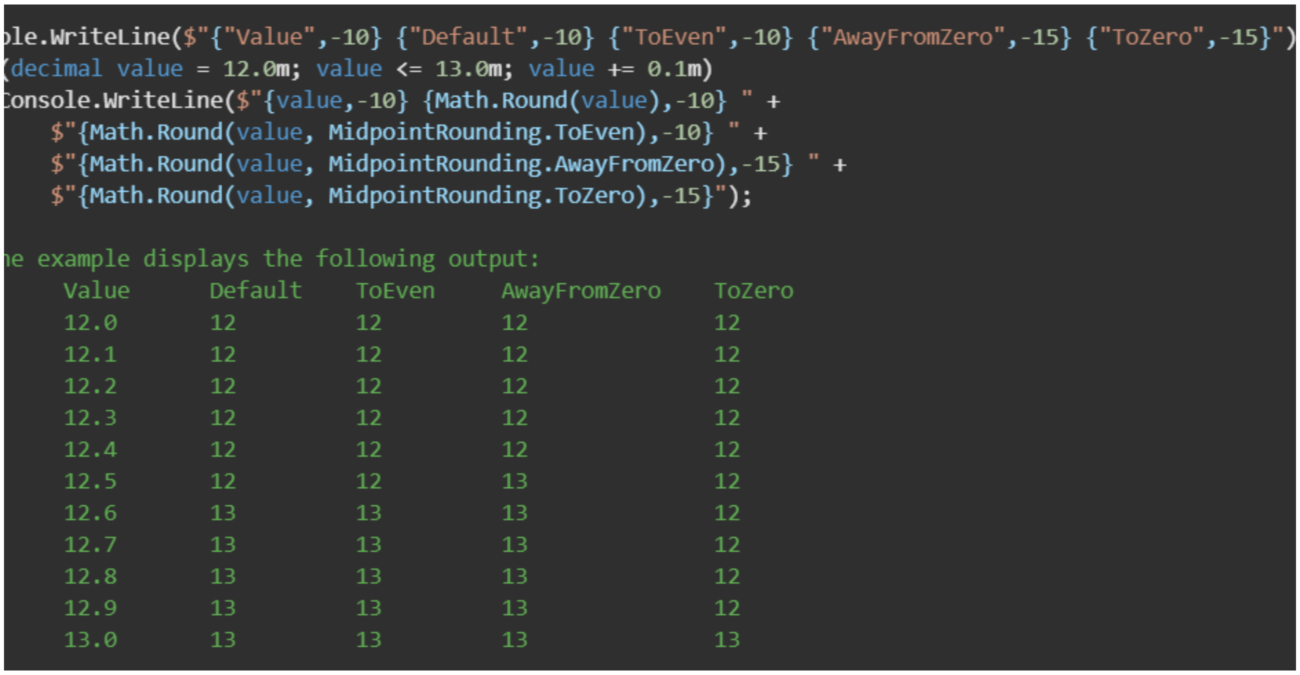
```
function difference(a, b) {
  return Math.abs(a - b);
}


console.log(difference(3, 5));
// expected output: 2


console.log(difference(5, 3));
// expected output: 2


console.log(difference(1.23456, 7.89012));
// expected output: 6.6555599999999995
```

23)Rounding off(roundf)

Rounds a value to the nearest integer or to the specified number of fractional digits.



Round(Double, Int32, MidpointRounding)	Rounds a double-precision floating-point value to a specified number of fractional digits using the specified rounding convention.
Round(Decimal, Int32, MidpointRounding)	Rounds a decimal value to a specified number of fractional digits using the specified rounding convention.
Round(Double, MidpointRounding)	Rounds a double-precision floating-point value to an integer using the specified rounding convention.
Round(Double, Int32)	Rounds a double-precision floating-point value to a specified number of fractional digits, and rounds midpoint values to the nearest even number.
Round(Decimal, Int32)	Rounds a decimal value to a specified number of fractional digits, and rounds midpoint values to the nearest even number.
Round(Double)	Rounds a double-precision floating-point value to the nearest integral value, and rounds midpoint values to the nearest even number.
Round(Decimal)	Rounds a decimal value to the nearest integral value, and rounds midpoint values to the nearest even number.
Round(Decimal, MidpointRounding)	Rounds a decimal value an integer using the specified rounding convention.



```
le.WriteLine($"{"Value",-10} {"Default",-10} {"ToEven",-10} {"AwayFromZero",-15} {"ToZero",-15}")
(decimal value = 12.0m; value <= 13.0m; value += 0.1m)
onsole.WriteLine($"{value,-10} {Math.Round(value),-10} " +
   $"{Math.Round(value, MidpointRounding.ToEven),-10} " +
   $"{Math.Round(value, MidpointRounding.AwayFromZero),-15} " +
   $"{Math.Round(value, MidpointRounding.ToZero),-15}");

e example displays the following output:
    Value      Default    ToEven     AwayFromZero    ToZero
    12.0       12         12         12              12
    12.1       12         12         12              12
    12.2       12         12         12              12
    12.3       12         12         12              12
    12.4       12         12         12              12
    12.5       12         12         13              12
    12.6       13         13         13              12
    12.7       13         13         13              12
    12.8       13         13         13              12
    12.9       13         13         13              12
    13.0       13         13         13              13
```

24)Rounding up (ceil)

Returns

The closest integer greater than or equal to *x*.

Description

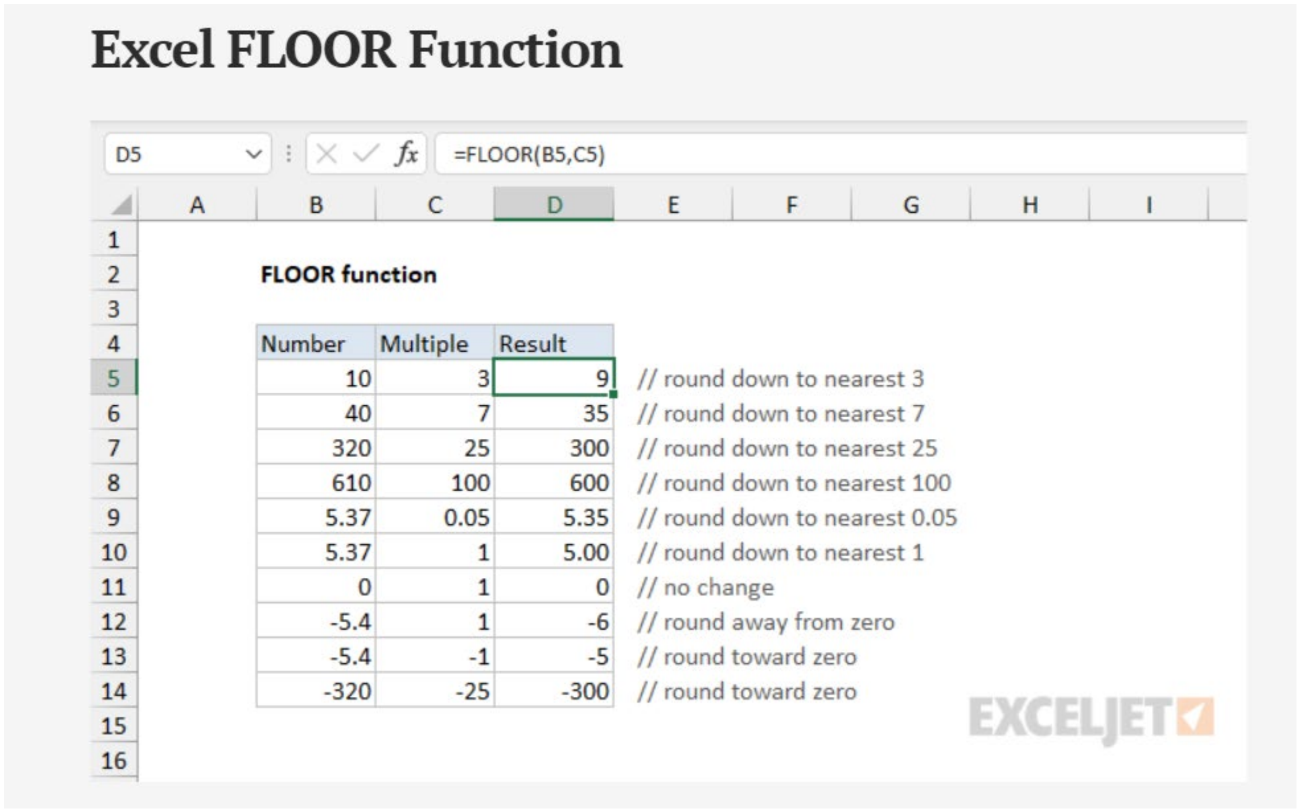
`Math.ceil( )` computes the ceiling function—i.e., it returns the closest integer value that is greater than or equal to the function argument. `Math.ceil( )` differs from `Math.round( )` in that it always rounds up, rather than rounding up or down to the closest integer. Also note that `Math.ceil( )` does not round negative numbers to larger negative numbers; it rounds them up toward zero.

Example

```
a = Math.ceil(1.99);   // Result is 2.0

b = Math.ceil(1.01);   // Result is 2.0

c = Math.ceil(1.0);    // Result is 1.0
```

```
d = Math.ceil(-1.99);  // Result is -1.0
```

25)Rounding down (floor)



Excel FLOOR Function

D5 =FLOOR(B5,C5)

FLOOR function

Number	Multiple	Result	
10	3	9	// round down to nearest 3
40	7	35	// round down to nearest 7
320	25	300	// round down to nearest 25
610	100	600	// round down to nearest 100
5.37	0.05	5.35	// round down to nearest 0.05
5.37	1	5.00	// round down to nearest 1
0	1	0	// no change
-5.4	1	-6	// round away from zero
-5.4	-1	-5	// round toward zero
-320	-25	-300	// round toward zero

26)Pyth.Thm. (a^2+b^2+c^2)

Pythagorean Theorem: a2 + b2 = c2

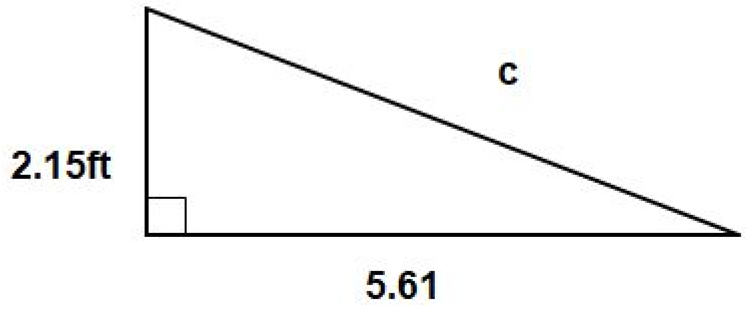
1. Find c.

$c^2 = 2.15^2 + 5.61^2$

$c^2 = 4.6225 + 31.4721$

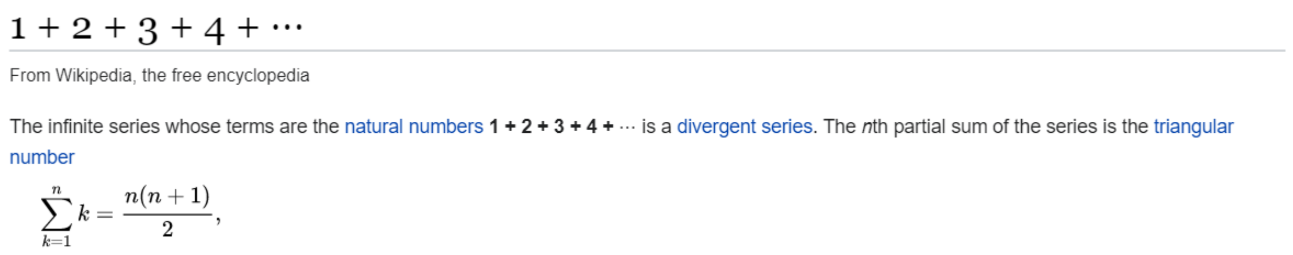
$c^2 = 36.0946$

c = ____



27)Floating Point Arithmetic (how floating points work)
https://en.wikipedia.org/wiki/IEEE_754

28)Arithmetic sequence (1,2,3)



$1 + 2 + 3 + 4 + \cdots$

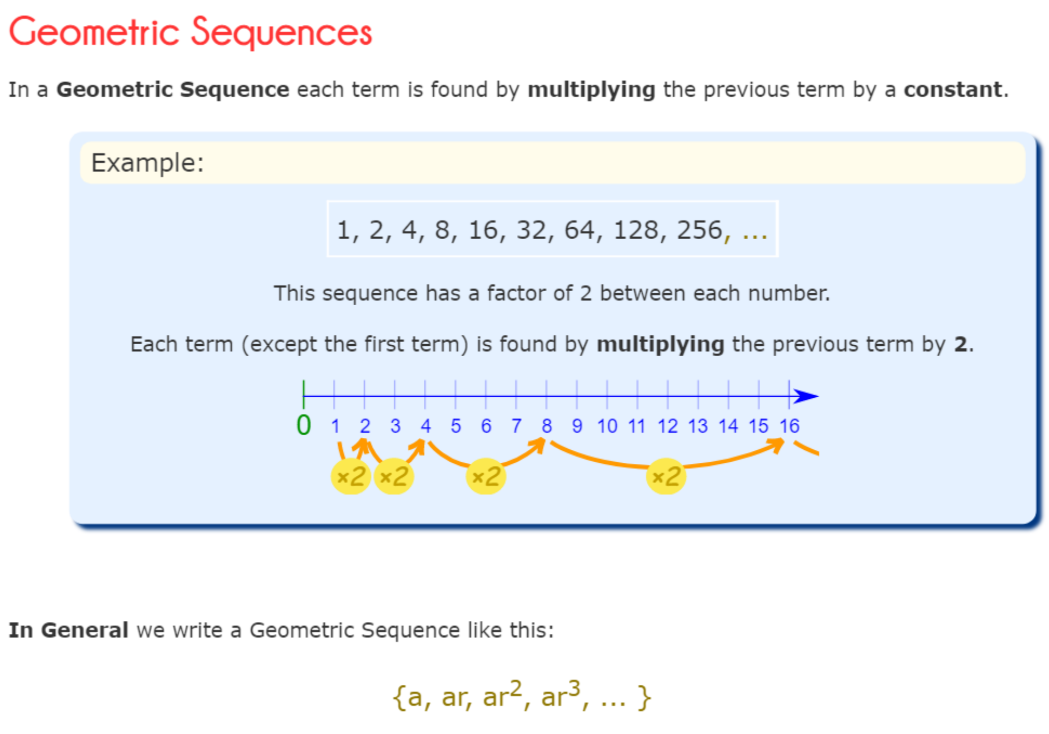
From Wikipedia, the free encyclopedia

The infinite series whose terms are the natural numbers **1 + 2 + 3 + 4 +** ⋯ is a divergent series. The *n*th partial sum of the series is the triangular number

$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2},$$

```
// Java program to check if a given array
// can form arithmetic progression
import java.util.Arrays;

class GFG {

    // Returns true if a permutation of
    // arr[0..n-1] can form arithmetic
    // progression
    static boolean checkIsAP(int arr[], int n)
    {
        if (n == 1)
            return true;

        // Sort array
        Arrays.sort(arr);

        // After sorting, difference between
        // consecutive elements must be same.
        int d = arr[1] - arr[0];
        for (int i = 2; i < n; i++)
            if (arr[i] - arr[i - 1] != d)
                return false;
        return true;
    }

    // driver code
    public static void main(String[] args)
    {
        int arr[] = { 20, 15, 5, 0, 10 };
        int n = arr.length;

        if (checkIsAP(arr, n))
            System.out.println("Yes");
        else
            System.out.println("No");
    }
}
```

29)Geometric sequence (1^2,2^2,3^2)



Geometric Sequences

In a **Geometric Sequence** each term is found by **multiplying** the previous term by a **constant**.

Example:

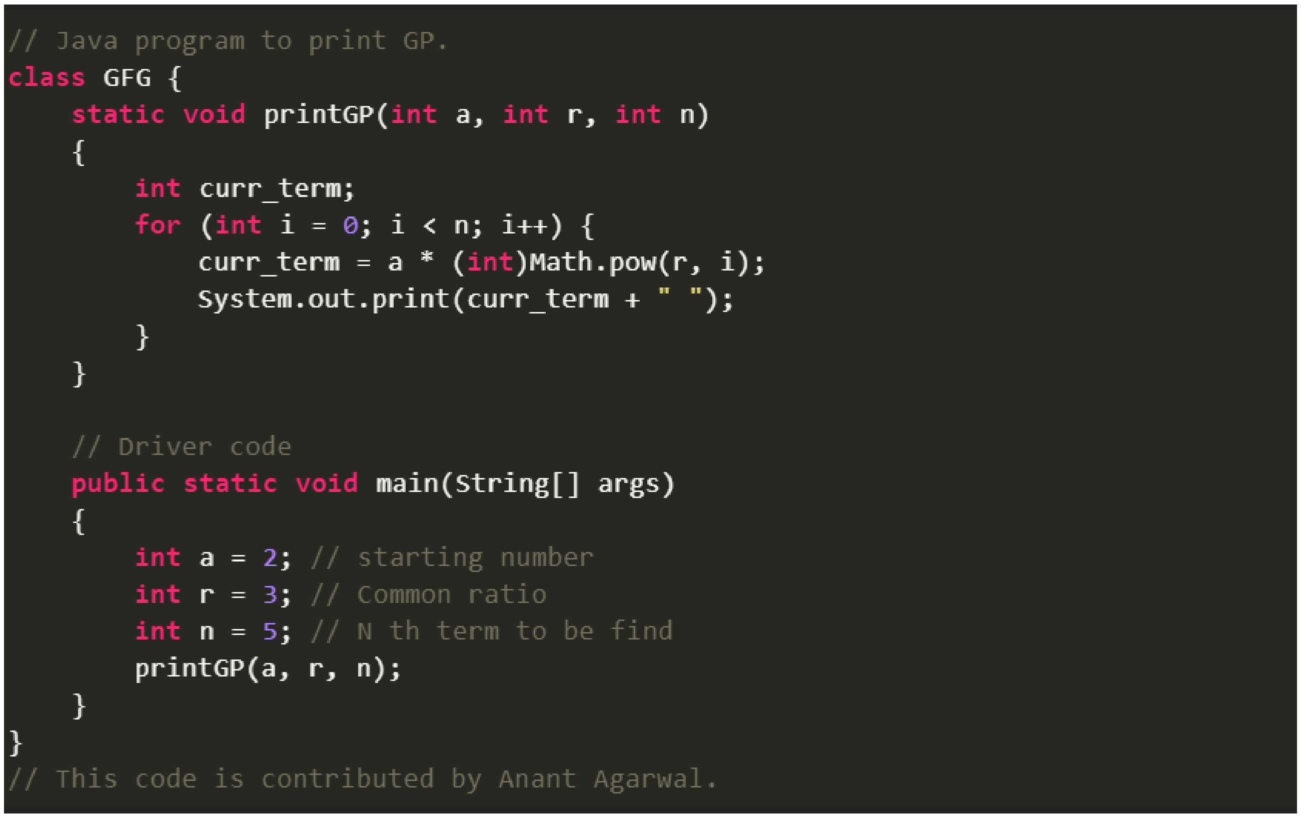
1, 2, 4, 8, 16, 32, 64, 128, 256, ...

This sequence has a factor of 2 between each number.

Each term (except the first term) is found by **multiplying** the previous term by **2**.

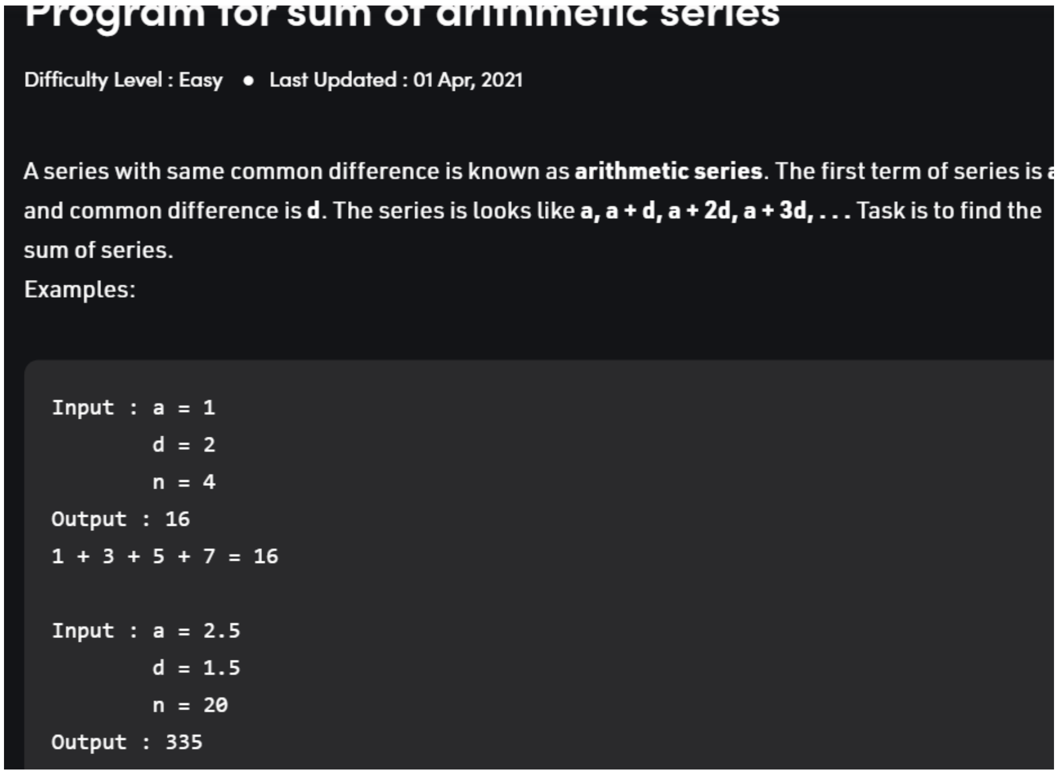
In General we write a Geometric Sequence like this:

$\{a, ar, ar^2, ar^3, \dots \}$



```java
// Java program to print GP.
class GFG {
    static void printGP(int a, int r, int n)
    {
        int curr_term;
        for (int i = 0; i < n; i++) {
            curr_term = a * (int)Math.pow(r, i);
            System.out.print(curr_term + " ");
        }
    }

    // Driver code
    public static void main(String[] args)
    {
        int a = 2; // starting number
        int r = 3; // Common ratio
        int n = 5; // N th term to be find
        printGP(a, r, n);
    }
}
// This code is contributed by Anant Agarwal.
```

<u>**30)How to do a sum of numbers (foe loop/arithmetic sequence)**</u>



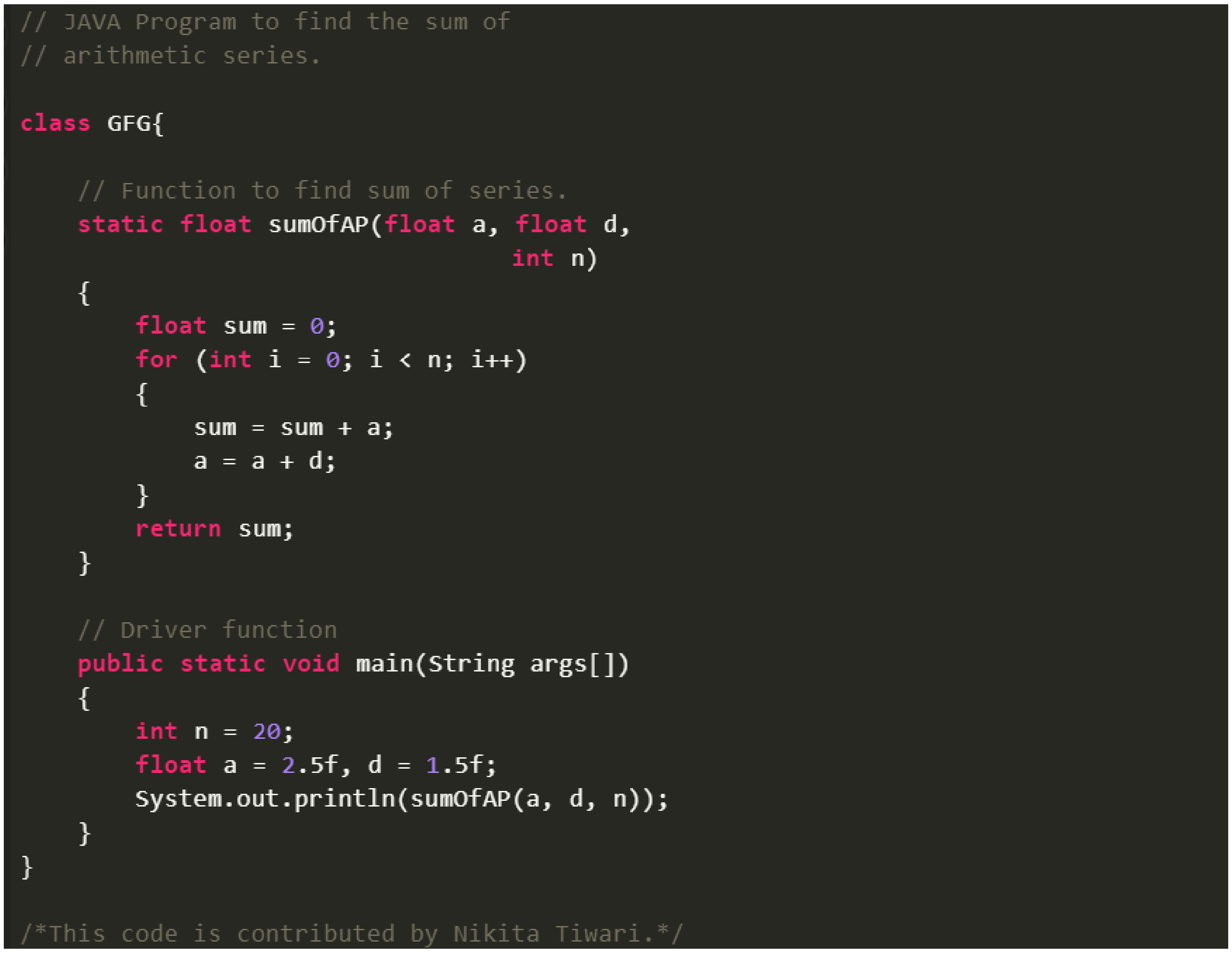
Program for sum of arithmetic series

Difficulty Level : Easy • Last Updated : 01 Apr, 2021

A series with same common difference is known as **arithmetic series**. The first term of series is and common difference is **d**. The series is looks like **a, a + d, a + 2d, a + 3d, . . .** Task is to find the sum of series.

Examples:

```
Input : a = 1
        d = 2
        n = 4
Output : 16
1 + 3 + 5 + 7 = 16

Input : a = 2.5
        d = 1.5
        n = 20
Output : 335
```



```java
// JAVA Program to find the sum of
// arithmetic series.

class GFG{

    // Function to find sum of series.
    static float sumOfAP(float a, float d,
                                  int n)
    {
        float sum = 0;
        for (int i = 0; i < n; i++)
        {
            sum = sum + a;
            a = a + d;
        }
        return sum;
    }

    // Driver function
    public static void main(String args[])
    {
        int n = 20;
        float a = 2.5f, d = 1.5f;
        System.out.println(sumOfAP(a, d, n));
    }
}

/*This code is contributed by Nikita Tiwari.*/
```

31)Binary Numbers

1011101111 - use calculator

32)Hexadecimal Numbers

The hexadecimal numeral system, often shortened to "hex", is a **numeral system made up of 16 symbols (base 16) 0-15**

Use calculator

33)Octal Numbers (8)

A number system **with its base as eight** and uses digits from 0 to 7 is called Octal Number System. The word octal is used to represent the numbers that have eight as the base. The octal numbers have many applications and importance such as it is used in computers and digital numbering systems.

Use Calculator

34)Factorial

$$n! = n \times (n-1) \times \cdots \times 1$$

```
public BigInteger factorialHavingLargeResult(int n) { BigInteger result =
BigInteger.ONE; for (int i = 2; i <= n; i++) result =
result.multiply(BigInteger.valueOf(i)); return result; }
```

35)Functions

https://www.javatpoint.com/java-math

36)Recursive Functions (Fibonacci sequence, Factorial of a number)

https://www.javatpoint.com/recursion-in-java

37)Maximum value/Upper limit, Lower limit of certain bytes of integer (32 vs 64 bits)

Signed and Unsigned Integers

The XDR standard defines signed integers as *integer*. A signed integer is a 32-bit datum that encodes an integer in the range [-2147483648 to 2147483647]. An unsigned integer is a 32-bit datum that encodes a nonnegative integer in the range [0 to 4294967295].

The signed integer is represented in twos complement notation. The most significant byte is 0 and the least significant is 3.

The unsigned integer is represented by an unsigned binary number whose most significant byte is 0; the least significant is 3. See the Signed Integer and Unsigned Integer figure (Figure 1).

```
public class Test { public static void main(String[] args) {
System.out.println(Integer.MIN_VALUE); System.out.println(Integer.MAX_VALUE); } }
```

How much can 64-bit handle?

In principle, a 64-bit microprocessor can address 16 EiB ($16 \times 1024^6 = 2^{64}$ = 18,446,744,073,709,551,616 bytes, or about 18.4 exabytes) of memory. However, not all instruction sets, and not all processors implementing those instruction sets, support a full 64-bit virtual or physical address space.

38)Floating point exceptions (Epsilon, division by 0, Infinity)

Floating-point exceptions

This topic provides information about floating-point exceptions and how your programs can detect and handle them.

The Institute of Electrical and Electronics Engineers (IEEE) defines a standard for floating-point exceptions called the IEEE Standard for Binary Floating-Point Arithmetic (IEEE 754). This standard defines five types of floating-point exception that must be signaled when detected:

- Invalid operation
- Division by zero
- Overflow
- Underflow
- Inexact calculation

When one of these exceptions occurs in a user process, it is signaled either by setting a flag or taking a trap. By default, the system sets a status flag in the Floating-Point Status and Control registers (FPSCR), indicating the exception has occurred. Once the status flags are set by an exception, they are cleared only when the process clears them explicitly or when the process ends. The operating system provides subroutines to query, set, or clear these flags.

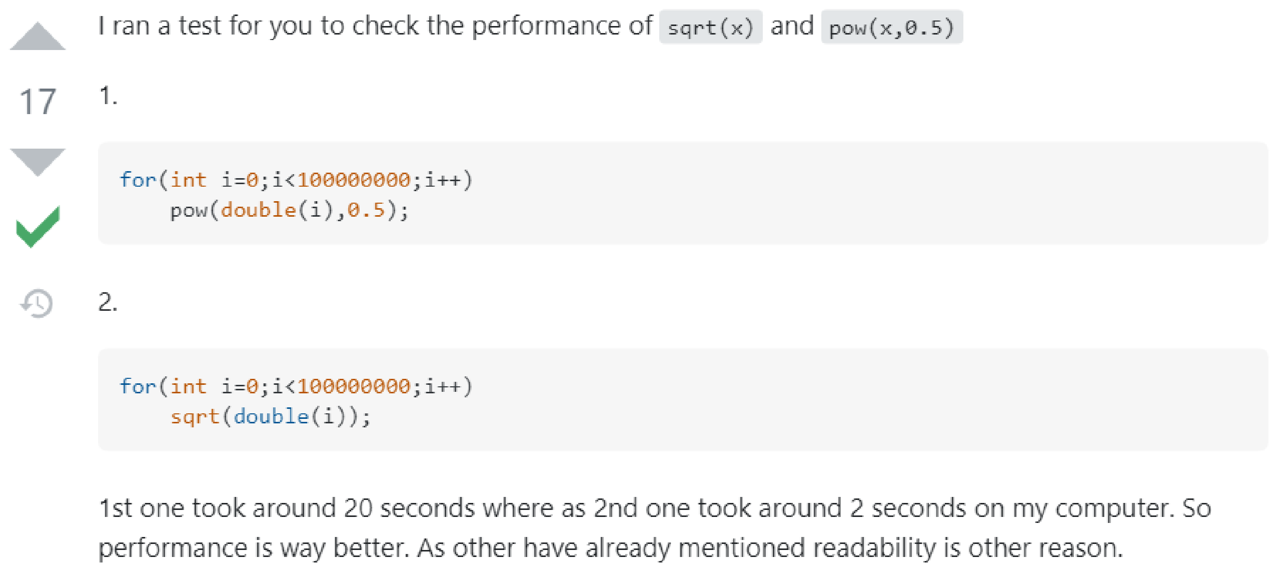
The system can also cause the floating-point exception signal (**SIGFPE**) to be raised if a floating-point exception occurs. Because this is not the default behavior, the operating system provides subroutines to change the state of the process so the signal is enabled. When a floating-point exception raises the **SIGFPE** signal, the process terminates and produces a core file if no signal-handler subroutine is present in the process. Otherwise, the process calls the signal-handler subroutine.

Floating-point exception subroutines

Floating-point exception subroutines can be used to:

- Change the execution state of the process
- Enable the signaling of exceptions
- Disable exceptions or clear flags
- Determine which exceptions caused the signal
- Test the exception sticky flags
- **Floating-point trap handler operation**
- To generate a trap, a program must change the execution state of the process using the **fp_trap** subroutine and enable the exception to be trapped using the **fp_enable** or **fp_enable_all** subroutine.
- Changing the execution state of the program may slow performance because floating-point trapping causes the process to execute in serial mode.
- When a floating-point trap occurs, the **SIGFPE** signal is raised. By default, the **SIGFPE** signal causes the process to terminate and produce a core file. To change this behavior, the program must establish a signal handler for this signal. See the **sigaction**, **sigvec**, or **signal** subroutines for more information on signal handlers.

39)Square Root(sqrt/pow(x,1/2)



I ran a test for you to check the performance of `sqrt(x)` and `pow(x,0.5)`

1.

```
for(int i=0;i<100000000;i++)
    pow(double(i),0.5);
```

2.

```
for(int i=0;i<100000000;i++)
    sqrt(double(i));
```

1st one took around 20 seconds where as 2nd one took around 2 seconds on my computer. So performance is way better. As other have already mentioned readability is other reason.

40)String (A set of 8bit numbers)

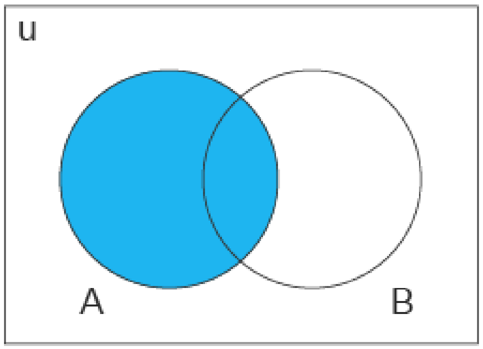
UTF-8 is a variable-width character encoding used for electronic communication. Defined by the Unicode Standard, the name is derived from *Unicode* (or *Universal Coded Character Set*) *Transformation Format – 8-bit*.[1]

41) Set

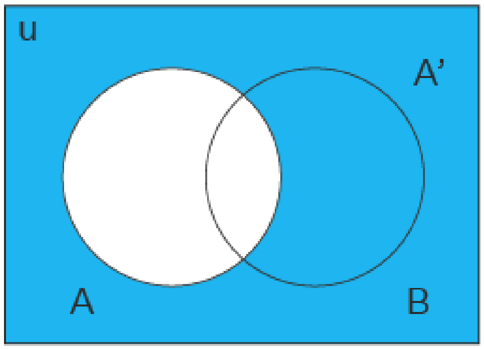
Set operations is **a concept similar to fundamental operations on numbers**. Sets in math deal with a finite collection of objects, be it numbers, alphabets, or any real-world objects. Sometimes a necessity arises wherein we need to establish the relationship between two or more sets.

Set Operations

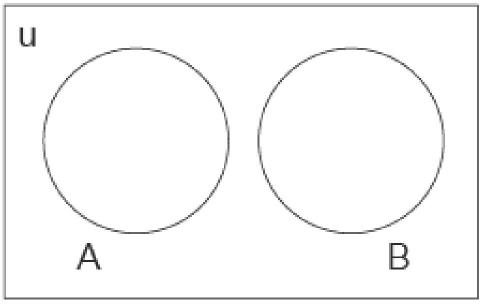
cuemath
THE MATH EXPERT



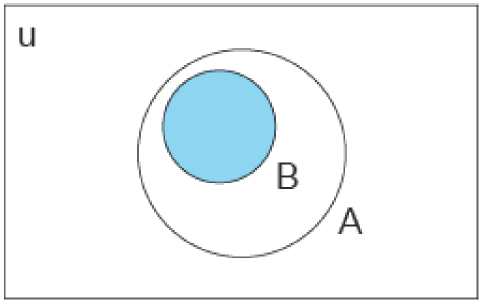
Set A



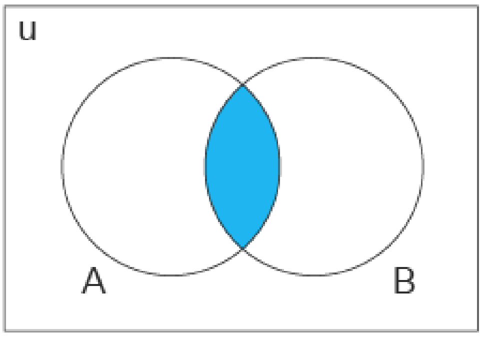
A' the complement of A



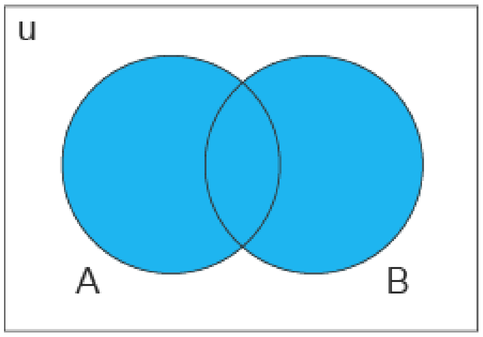
A and B are disjoint sets



B is proper subset of A $B \subset A$



Both A and B
A intersect B $A \cap B$



Either A or B
A union B $A \cup B$