```
#define cin_2d(vec, n, m) for(int i = 0; i < n; i++) for(int j = 0; j < m && cin >> vec[i][j]; j++);

#define cout_2d(vec, n, m) for(int i = 0; i < n; i++, cout << "\n") for(int j = 0; j < m && cout << vec[i][j] << " "; j++);

#define fixed(n) fixed << setprecision(n)

#define ceil(n, m) (((n) / (m)) + ((n) % (m) ? 1 : 0))

#define fill(vec, value) memset(vec, value, sizeof(vec));

#define mul_mod(a, b, m) (((a % m) * (b % m)) % m)

#define add_mod(a, b, m) (((a % m) + (b % m)) % m)

#define all(vec) vec.begin(), vec.end()

#define rall(vec) vec.rbegin(), vec.rend()

#define sz(x) int(x.size())

#define debug(x) cout << #x << ": " << (x) << "\n";

#define fi first

#define se second

#define ll long long

#define ull unsigned long long

#define Mod  1'000'000'007

#define OO 2'000'000'000

#define EPS 1e-9

#define PI acos(-1)

template < typename T = int > using Pair = pair < T, T >;

vector < string > RET = {"NO", "YES"};

template < typename T = int > istream& operator >> (istream &in, vector < T > &v) {
   for (auto &x : v) in >> x;
   return in;
}

template < typename T = int > ostream& operator << (ostream &out, const vector < T > &v) {
   for (const T &x : v) out << x << ' ';
   return out;
```

```
}
    // Greatest common divisors between two numbers
    ll GCD(ll a, ll b){
        return (!b ? a : GCD(b, a % b));
    }
    // least common multiplication between two numbers
    ll LCM(ll a, ll b){
        return a / GCD(a, b) * b;
    }
    // Combination
    ll nCr(ll n, ll r){
        if(r > n) return 0;
        ll p = 1, k = 1;
        if (n - r < r) r = n - r;
        // condition for minimum choose
        if(n < 1) return 0;
        while (r > 0){
            p *= n, k *= r;
            ll m = __gcd(p, k);
            p /= m, k /= m, n--, r--;
        }
        return p;
    }
    // Permutation
    ll nPr(ll n, ll r){
        if(r > n) return 0;
        ll npr = 1;
        while(r-- > 0)
            npr *= n--;
```

```
    return npr;
}
// get a mod for big int
ll Big_Mod(string s, ll mod){
    ll res = 0;
    for(auto& c : s)
        res = (res * 10 + (c - '0')) % mod;
    return res;
}
// add two number and take mod for them
void add(ll& a, ll b, ll mod = 1e9 + 7){
    a += b;
    if(a >= mod)
        a -= mod;
}
// multiply two number and take mod for them
void mul(ll& a, ll b, ll mod = 1e9 + 7){
    a = ((a % mod) * (b % mod)) % mod;
}
// Check if number is prime or not
bool is_prime(ll n){
    if(n < 2 || (n % 2 == 0 && n != 2)) return false;
    for(int i = 3; i <= sqrt(n); i += 2)
        if(n % i == 0) return false;
    return true;
}
// get the number of divisors for n
int number_of_divisors(ll n){
    int divisors = 0;
```

```
    for(int i = 1; i < sqrt(n); i++)
        if(n % i == 0) divisors += 2;
    return divisors + (sqrt(n) == (int)sqrt(n));
}
// get Summation of divisors for n
ll sum_of_divisors(ll n){
    ll sum_divisors = 0;
    for(int i = 1; i < sqrt(n); i++)
        if(n % i == 0) sum_divisors += ((n / i) + i);
    ll sq = sqrt(n);
    return sum_divisors + (sq * sq == n ? sq : 0);
}
// sum of divisor of number in range [1 ... n]
ll divisorSum(ll num){
    ll sum = 0;
    for (ll i = 1; i <= sqrt(num); i++) {
        ll t1 = i * (num / i - i + 1);
        ll t2 = (((num / i) * (num / i + 1)) / 2) - ((i * (i + 1)) / 2);
        sum += t1 + t2;
    }
    return sum;
}
// get vector with the divisors for n
vector < ll > Get_Divisors(ll n){
    vector < ll > divisors;
    for(int i = 1; i < sqrt(n); i++)
        if(n % i == 0) divisors.push_back(i), divisors.push_back(n / i);
    if(sqrt(n) == int(sqrt(n))) divisors.push_back(sqrt(n));
    return divisors;
```

```
}
// print all permutation of an array
void Print_Permutation(vector < int >& nums){
    sort(all(nums));
    do {
        for(auto& i : nums)
            cout << i << " ";
        cout << "\n";
    } while(next_permutation(nums.begin(), nums.end()));
}
// print all permutation of a string
void Print_Permutation(string s){
    sort(all(s));
    do {
        cout << s << "\n";
    } while(next_permutation(s.begin(), s.end()));
}
// get the summation between two numbers or the summation between 1 and n
ll Summation(ll r, ll l = 0){
    if(l > r) swap(l, r);
    return (r * (r + 1) / 2) - (l * (l - 1) / 2);
}
// Get how many number divisable by c between a and b
ll how_many_divisors(ll a, ll b, ll c){
    return (b / c) - ((a - 1) / c);
}
// Get summation of numbers divisable by c between a and b
ll Summation_of_Devisors(ll a, ll b, ll c){
    ll right = Summation(b / c);
```

```
    ll left = Summation((a - 1) / c);
    return (right - left) * c;
  }
  // get logb(a)
  double get_log(ll a, int b){
    return log(a) / log(b);
  }
  // Check if number power of another or not
  bool is_power(ll number, int base = 2){
    return (get_log(number, base) - (ll) get_log(number, base) <= EPS);
  }
  // Distination Between two points
  double dist(double x1, double y1, double x2, double y2){
    return sqrt(pow(x1 - x2, 2) + pow(y1 - y2, 2));
  }
  // Check if it valid triangle with 3 length sides
  bool is_triangle(ll a, ll b, ll c){
    return (a + b > c) && (a + c > b) && (b + c > a) && (a && b && c);
  }
  // Get Slope of two points
  double slope(double x1, double y1, double x2, double y2){
    if(x2 == x1) return 0;
    return (y2 - y1) / (x2 - x1);
  }
 // Check if three points in the same line
bool is_same_line(ll x1, ll y1, ll x2, ll y2, ll x3, ll y3){
    return (y2 - y1) * (x3 - x1) == (y3 - y1) * (x2 - x1);
  }
  // Check if is perfect square
```

```
    bool is_perfect_square(ll n){
        ll sq = sqrt(n);
        return sq * sq == n;
    }
    // get the power of prime factor in n
    ll FactN_PrimePowers(ll n, ll p){
        ll powers = 0;
        for(ll i = p; i <= n; i *= p)
            powers += n / i;
        return powers;
    }
// extended euclidean algorithm and diofantian equation
    int extended_gcd(int a, int b, int& x, int& y) {
        if (b == 0) {
            x = 1;
            y = 0;
            return a;
        }
        int x1, y1;
        int d = extended_gcd(b, a % b, x1, y1);
        x = y1;
        y = x1 - y1 * (a / b);
        return d;
    }
    // Convert Decimal to any base
    string decimal_to_any_base(ll decimal, ll base){
        if(decimal == 0) return "0";
        string num = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";
        string result;
```

```
        do{
            result.push_back(num[decimal % base]);
            decimal /= base;
        }while(decimal != 0);
        return string(result.rbegin(), result.rend());
    }
    // Convert any base to decimal
    ll  any_base_to_decimal(string str, int base) {
        auto val = [](char c){
            return (c >= '0' && c <= '9' ? (int) c - '0' : (int) c - 'A' + 10);
        };
        ll len = sz(str), power = 1, num = 0, i;
        for (i = len - 1; i >= 0; i--) {
            num += val(str[i]) * power;
            power = power * base;
        }
        return num;
    }
};
```

//snippets

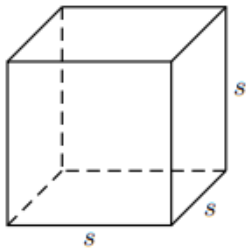
```
/////sieve_lp_hp_pf
int N=1e7+10;
vector<bool> isPrime(N,true);
vector<int> lp(N,0);//lowest prime of n
vector<int> hp(N,0);//highest prime of n
void sieveNorm(){//O(n*log(log(N)))
    isPrime[0]=isPrime[1]=false;
    for(int i=2; i<N; i++){
```

```
        if(isPrime[i]==true){
            lp[i]=hp[i]=i;
            for(int j=i+i; j<N; j+=i){
                isPrime[j]=false;
                hp[j]=i;
                if(lp[j]==0)
                    lp[j]=i;
            }
        }
    }
}
vector<int> p_freq;
void p_factorization(long long val){//O(logN)
    while(val>1){
        int pf=hp[val];
        while(val%pf==0){
            val/=pf;
            p_freq.push_back(pf);
        }
    }
}
void lineiss(){
    string s;
    getline(cin,s);
    istringstream is(s);
    int x;
    string a;
    is>>a>>x;
    cout<<a<<endl;
```

```
    cout<<x<<endl;
}
map<long long,long long> freq;
void factorization_of_factorial(long long val){
    int i=0;
    ll ans=1;
    while(prime[i]<=val){
        ll x=0;
        ll t=floor(log2(val)/log2(prime[i]));
        for(ll j=1;j<=t;j++){
            ll z=floor(val/pow(prime[i],j));
            x+=z;
        }
        freq[prime[i]]=x;
        i++;
    }
}
set<long long> factors_of_N(long long n){//O(sqrt(N)))
    set<long long> facts;
    facts.insert(1);
    facts.insert(n);
    for(long long i=1; i*i<=n; i++){
        if(n%i==0){
            facts.insert(i);
            facts.insert(n/i);
        }
    }
    return facts;
}
```

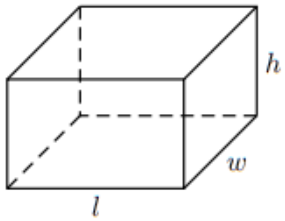
CUBE

s = side
Volume: $V = s^3$
Surface Area: $S = 6s^2$



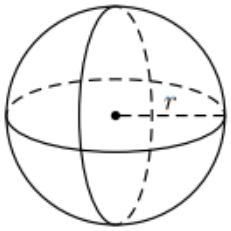
RECTANGULAR SOLID

l = length, w = width,
h = height
Volume: $V = lwh$
Surface Area:
$S = 2lw + 2lh + 2wh$



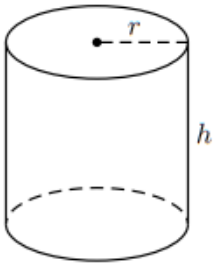
SPHERE

r = radius
Volume: $V = \frac{4}{3}\pi r^3$
Surface Area: $S = 4\pi r^2$



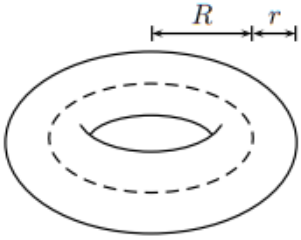
RIGHT CIRCULAR CYLINDER

r = radius, h = height
Volume: $V = \pi r^2 h$
Surface Area: $S = 2\pi rh + 2\pi r^2$



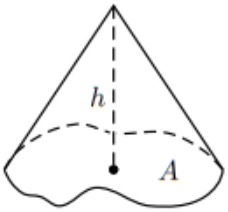
TORUS

r = tube radius,
R = torus radius
Volume: $V = 2\pi^2 r^2 R$
Surface Area: $S = 4\pi^2 rR$



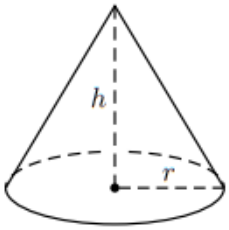
GENERAL CONE OR PYRAMID

A = area of base, h = height
Volume: $V = \frac{1}{3}Ah$



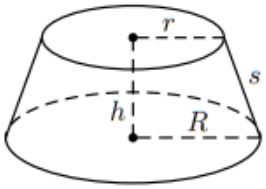
RIGHT CIRCULAR CONE

r = radius, h = height
Volume: $V = \frac{1}{3}\pi r^2 h$
Surface Area:
$S = \pi r\sqrt{r^2 + h^2} + \pi r^2$



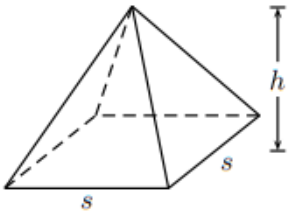
FRUSTUM OF A CONE

r = top radius, R = base radius,
h = height, s = slant height
Volume: $V = \frac{\pi}{3}(r^2 + rR + R^2)h$
Surface Area:
$S = \pi s(R + r) + \pi r^2 + \pi R^2$



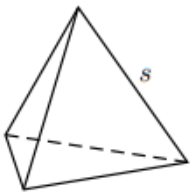
SQUARE PYRAMID

s = side, h = height
Volume: $V = \frac{1}{3}s^2 h$
Surface Area:
$S = s(s + \sqrt{s^2 + 4h^2})$



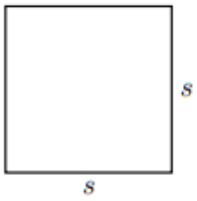
REGULAR TETRAHEDRON

s = side
Volume: $V = \frac{1}{12}\sqrt{2}s^3$
Surface Area: $S = \sqrt{3}s^2$



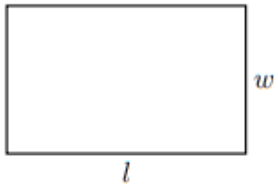
SQUARE

s = side
Area: $A = s^2$
Perimeter: $P = 4s$



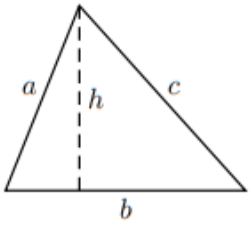
RECTANGLE

l = length, w = width
Area: $A = lw$
Perimeter: $P = 2l + 2w$



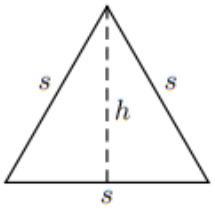
TRIANGLE

b = base, h = height
Area: $A = \frac{1}{2}bh$
Perimeter: $P = a + b + c$



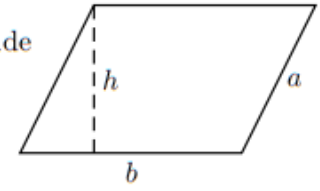
EQUILATERAL TRIANGLE

s = side
Height: $h = \frac{\sqrt{3}}{2}s$
Area: $A = \frac{\sqrt{3}}{4}s^2$



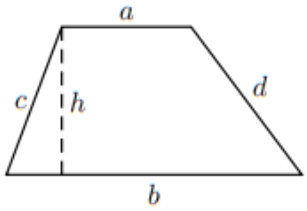
PARALLELOGRAM

b = base, h = height, a = side
Area: $A = bh$
Perimeter: $P = 2a + 2b$



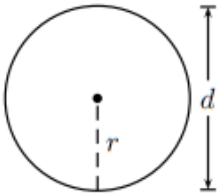
TRAPEZOID

a, b = bases; h = height;
c, d = sides
Area: $A = \frac{1}{2}(a + b)h$
Perimeter:
$P = a + b + c + d$



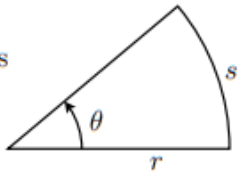
CIRCLE

r = radius, d = diameter
Diameter: $d = 2r$
Area: $A = \pi r^2$
Circumference: $C = 2\pi r = \pi d$



SECTOR OF CIRCLE

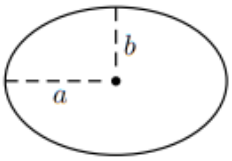
r = radius, θ = angle in radians
Area: $A = \frac{1}{2}\theta r^2$
Arc Length: $s = \theta r$



ELLIPSE

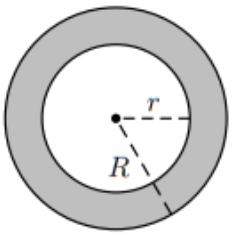
a = semimajor axis
b = semiminor axis
Area: $A = \pi ab$

Circumference:
$C \approx \pi \left(3(a + b) - \sqrt{(a + 3b)(b + 3a)}\right)$



ANNULUS

r = inner radius,
R = outer radius
Average Radius: $\rho = \frac{1}{2}(r + R)$
Width: $w = R - r$
Area: $A = \pi(R^2 - r^2)$
or $A = 2\pi\rho w$



REGULAR POLYGON

s = side length,
n = number of sides
Circumradius: $R = \frac{1}{2}s\csc(\frac{\pi}{n})$
Area: $A = \frac{1}{4}ns^2\cot(\frac{\pi}{n})$
or $A = \frac{1}{2}nR^2\sin(\frac{2\pi}{n})$