Arif Template

```
——————————————————————————————

/* "بِسْمِ اللهِ الرَّحْمٰنِ الرَّحِيْمِ " -In the name of Allah."
---------- A R I F ---------- */

#include<bits/stdc++.h>
// for order set ----------
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
#define ordered_set tree<ll, null_type, less<ll>,
rb_tree_tag, tree_order_statistics_node_update>
#define index_of(x) find_by_order(x) // index_of the
value x
#define number_of(x) order_of_key(x) // how many
value are stricly less then x
// ------------------------------------
#define int long long int
#define INF 1e18
#define PI 3.141592653
#define PB push_back
#define F first
#define S second
#define MP(x, y) push_back(make_pair(x, y))
#define srt(v) sort(v.begin(), v.end())
#define all(x) x.begin(), x.end()
#define rsrt(v) reverse(v.begin(), v.end())
#define no cout << "NO" << endl
#define yes cout << "YES" << "\n"
#define e "\n"
#define pair vector< pair <int ,int> >
#define deb(args...){string _s =
#args;replace(_s.begin(), _s.end(), ',', ' ');stringstream
_ss(_s);istream_iterator<string> _it(_ss);err(_it, args);}

using namespace std;

template <typename T>
ostream &operator<<(ostream &os, const vector<T>
&v){ os << '{'; for (const auto &x : v) os << " " << x;
return os << '}';}
void err(istream_iterator<string> it) {} template
<typename T, typename... Args>
void err(istream_iterator<string> it, T a, Args... args){
cerr << *it <<" = " << a << endl; err(++it, args...);}
/*--------------------------------------------------------------------------
---------------*/
int expo(int a, int b, int mod) {int res = 1; while (b > 0) {if
(b & 1)res = (res * a) % mod; a = (a * a) % mod; b = b
>> 1;} return res;}
int mod_add(int a, int b, int m) {a = a % m; b = b % m;
return (((a + b) % m) + m) % m;}
int mod_mul(int a, int b, int m) {a = a % m; b = b % m;
return (((a * b) % m) + m) % m;}
int mod_sub(int a, int b, int m) {a = a % m; b = b % m;
return (((a - b) % m) + m) % m;}
int mminvprime(int a, int b) {return expo(a, b - 2, b);}
int mod_div(int a, int b, int m) {a = a % m; b = b % m;
return (mod_mul(a, mminvprime(b, m), m) + m) % m;}
//only for prime m
mt19937
rng(chrono::steady_clock::now().time_since_epoch().co
unt());
int getRandomNumber(int l, int r) {return
uniform_int_distribution<int>(l, r)(rng);}
bool isBitSet(int num, int bitPosition) { return (num & (1
<< bitPosition)) > 0 ; }
/*--------------------------------------------------------------------------
---------------*/
void solve()
{
    int n = 0 , m = 0 , k = 0 , ans = 0 , cnt = 0 ;

}
int32_t main()
{

    ios_base::sync_with_stdio(false);cin.tie(NULL);
    int test_case =1;
    cin >> test_case ;
    int c = 0 ;
    while( test_case --)
    {
      // c ++ ; cout << "Case " << c << ": " ;
       solve() ;
    }
}
//vector<int>::iterator lower, upper;
//lower = lower_bound(v.begin(), v.end(), value) -
v.begin() ;
//upper = upper_bound(v.begin(), v.end(), value) -
v.begin() ; -->
// scanf("%s%d",s,&x) != EOF

//---------DFS----------
const int mx = 1e6+7 ;
vector<int> g[mx] ;
int vis[mx] = {0} ;
void graph(int u , int v )
{
    g[u].PB(v) ; g[v].PB(u) ;
}
void dfs(int node)
{
    vis[node] = 1 ;
    for(auto it : g[node])
        if(!vis[it])
            dfs(it) ;
}
void Clear(int n)
{
  for(int i =0 ; i<=n ; i++)
  {
    g[i].clear() ;
    vis[i] =0 ;
```

```
  }
}

—---------BFS
const int mx =100000+5 ;
vector<int> g[mx] ;
int vis[mx] = {0} ;
int dis[mx] = {0} ;
queue<int> q ;
void bfs(int node)
{
   vis[node] = 1 ;
   dis[node] = 0 ;
   q.push(node) ;
   while(!q.empty())
   {
      int tem = q.front() ; q.pop() ;
      for(auto it : g[tem])
      {
         if(!vis[it])
         {
            vis[it] = 1 ;
            q.push(it) ;
            dis[it] = dis[tem]+1 ;
         }
      }
   }
}
//----------prime factor
const int MAXN = 1e6 + 5;
vector<int> primes;
bool is_prime[MAXN];

void sieve_of_eratosthenes() {
   memset(is_prime, true, sizeof is_prime);
   for (int p = 2; p * p <= MAXN; p++) {
      if (is_prime[p] == true) {
         for (int i = p * p; i <= MAXN; i += p)
            is_prime[i] = false;
      }
   }
   for (int p = 2; p <= MAXN; p++)
      if (is_prime[p])
         primes.push_back(p);
}
//--------------------------------
//prime_factorization----------------------------------------------------
----------
vector<int> prime_factorization(int n)
{
   vector<int> factors;
   for (int i = 0; primes[i] <= n / primes[i] ; i++) {
      while (n % primes[i] == 0) {
         factors.push_back(primes[i]);
         n /= primes[i];
      }
   }
   if (n > 1) {
      factors.push_back(n);
   }
   return factors;
}
//---------DSU---------------------------------------------------------
-----
const int mx = 1e5+5 ;
int parent[mx] ;
void init(int n)
{
   for (int i = 1; i <= n; ++i) parent[i] = i ;
}
int Find(int u){
   if (u == parent[u]) return u;
   return parent[u] = Find(parent[u]);
}

void Union(int u, int v){
   int p = Find(u);
   int q = Find(v);
   if (p != q) parent[q] = p;
}

bool isFriend(int u, int v){
   return Find(u) == Find(v);
}

//--------Prefix sum 2d----------------------
int mx = Size of 2D array! ;
int pre[mx][mx] ;
pre[0][0] = arr[0][0] ;
for(int i =1 ; i<mx ; i++) pre[0][i] = arr[0][i]+pre[0][i-1] ;
for(int i =1 ; i<mx ; i++) pre[i][0] = arr[i][0]+pre[i-1][0] ;
for(int i =1 ; i<mx ; i++)
   for(int j = 1 ; j<mx ; j++)
      pre[i][j] = pre[i-1][j] + pre[i][j-1]+ arr[i][j] -
 pre[i-1][j-1] ;
//------- Prefix sum --------------------

//---------chechk palindrome--------------------
bool is_palindrome(int i , string &s)
{
   if(i >= s.size()/2) return true ;
   if(s[i] != s[s.size()-i-1]) return false ;
   return is_palindrome(i+1 , s) ;
}
//---------chechk palindrome--------------------

—------KMP—---------------------
std::vector<int> lpsarraygenerate(string s)
{
   std::vector<int> lps(s.size());
   lps[0] = 0;  int j =0;
   for(int i = 1; i< s.size() ; i++)
   {
      while(j && s[j] != s[i])
         j = lps[j-1] ;
      if(s[i] == s[j]) j++ ;
```

```
        lps[i] = j ;
    }
    return lps ;
}

int KMP(const std::string& text, const std::string&
pattern) {
  int n = text.length();
  int m = pattern.length();
  int cnt = 0 ;
  std::vector<int> lps = lpsarraygenerate(pattern);
  int q = 0;
  for (int i = 0; i < n; ++i) {
    while (q > 0 && pattern[q] != text[i])
      q = lps[q - 1] ;
    if (pattern[q] == text[i]) q++ ;
    if (q == m) {
     cnt++ ;
    }
  }
  return cnt;
}
—---------NCR—----------
const int mx = 2e5 + 5;
int mod = INF ;
vector<int>f;

void fact() {
    int k = 1; f.PB(1);
    for (int i = 1; i <= mx; i++) {
        k *= i; k %= mod;
        f.PB(k);
    }
}

int power(int x, int y, int p) {
    int res = 1;
    x = x % p;
    while (y > 0) {
        if (y & 1)
            res = (res * x) % p;
        y = y >> 1;
        x = (x * x) % p;
    }
    return res;
}

int modInverse(int n, int p) {
    return power(n, p - 2, p);
}

int ncr(int n, int r) {
    if (n < r) return 0;
    if (r == 0) return 1;
    int p = mod;
    return (f[n] * modInverse(f[r], p) % p
            * modInverse(f[n - r], p) % p)
           % p;
}


//matix rotaion of 90----------
void rotateMatrix(vector<vector<int>>& matrix) {
    int m = matrix.size();
    int n = matrix[0].size();

    for (int i = 0; i < m/2; i++) {
        swap(matrix[i], matrix[m-i-1]);
    }

    for (int i = 0; i < m; i++) {
        for (int j = i+1; j < n; j++) {
            swap(matrix[i][j], matrix[j][i]);
        }
    }
}
// matix rotaion of 90 ----------

_____convex hull—----------

const int prec = 0.0000001;
struct point {
    ll x, y;
    point(ll xloc, ll yloc) : x(xloc), y(yloc) {}
    point() {}
    point& operator= (const point& other) {
        x = other.x, y = other.y;
        return *this;
    }
    bool operator== (const point& other) const {
        return abs(other.x - x) < prec && abs(other.y - y) <
prec;
    }
    bool operator!= (const point& other) const {
        return !(abs(other.x - x) < prec && abs(other.y - y) <
prec);
    }
    bool operator< (const point& other) const {
        return (x < other.x ? true : (x == other.x && y <
other.y));
    }
};
point p0;

int direction(point a, point b, point c) {
    int val = (b.y - a.y)*(c.x - b.x) - (b.x - a.x)*(c.y - b.y);
    return val == 0 ? 0 : (val < 0 ? 2 : 1); // 0: colinear, 1:
clockwise, 2: anti-clockwise
}

double squared_dist(point p1, point p2) {
    return (pow(p1.x - p2.x, 2) + pow(p1.y - p2.y, 2));
}

bool comparator(point p1, point p2) {
```

```
    int dir = direction(p0, p1, p2);
    if (dir == 0)
        return squared_dist(p0, p2) < squared_dist(p0, p1);
    return dir != 2;
}

point second_top(stack<point> &stack) {
    point top = stack.top(); stack.pop();
    point second = stack.top();
    stack.push(top);
    return second;
}

void graham_scan(vector<point> &points, int n) {
    int min_y = points[0].y;
    int min = 0;

    for (int i = 1; i < n; i++) {
        int y = points[i].y;

        // find bottom most or left most point
        if (y < min_y || y == min_y && points[i].x <
points[min].x) {
            min_y = points[i].y;
            min = i;
        }
    }

    swap(points[0], points[min]);
    p0 = points[0];
    sort(points.begin(), points.end(), comparator);
    p0 = points[0];

    stack<point> stack;
    stack.push(points[0]);
    stack.push(points[1]);

    for (int i = 2; i < n; i++) {
        while (stack.size() > 1 &&
direction(second_top(stack), stack.top(), points[i]) != 1){
            // point temp = stack.top() ; deb(temp.x , temp.y)
;
            stack.pop();
        }
        stack.push(points[i]);
    }

    cout << stack.size() << endl;
    // std::map<int, value> map;
    while (!stack.empty()) {
        point vertex = stack.top();
        cout << vertex.x << " " << vertex.y << endl;
        stack.pop();
    }
}

//-----------intersection check--------------
struct Point
{
    int x;
    int y;
};
// point q lies on line segment 'p-r'
bool onSegment(Point p, Point q, Point r)
{
    if (q.x <= max(p.x, r.x) && q.x >= min(p.x, r.x) &&
        q.y <= max(p.y, r.y) && q.y >= min(p.y, r.y))

        return true;

    return false;
}

int orientation(Point p, Point q, Point r)
{
    int val = (q.y - p.y) * (r.x - q.x) -
              (q.x - p.x) * (r.y - q.y);

    if (val == 0) return 0;  // collinear

    return (val > 0)? 1: 2; // clock or counterclock wise
}
bool doIntersect(Point p1, Point q1, Point p2, Point q2)
{
    int o1 = orientation(p1, q1, p2);
    int o2 = orientation(p1, q1, q2);
    int o3 = orientation(p2, q2, p1);
    int o4 = orientation(p2, q2, q1);

    if (o1 != o2 && o3 != o4)
        return true;

    if (o1 == 0 && onSegment(p1, p2, q1)) return true;
    if (o2 == 0 && onSegment(p1, q2, q1)) return true;
    if (o3 == 0 && onSegment(p2, p1, q2)) return true;
    if (o4 == 0 && onSegment(p2, q1, q2)) return true;
    return false; // Doesn't fall in any of the above cases
}
//-------------------------------------------------------
Lazy
struct info {
    i64 prop, sum;
} tree[mx * 3]; //sum ছাড়াও নিচে অতিরিক্ত কত যোগ হচ্ছে সেটা
রাখবো prop এ
void update(int node, int b, int e, int i, int j, i64 x)
{
    if (i > e || j < b)
        return;
    if (b >= i && e <= j) //নোডের রেঞ্জ আপডেটের রেঞ্জের ভিতরে
    {
        tree[node].sum += ((e - b + 1) * x); //নিচে নোড আছে
e-b+1 টি, তাই e-b+1 বার x যোগ হবে এই রেঞ্জে
        tree[node].prop += x; //নিচের নোডগুলোর সাথে x যোগ
হবে
        return;
    }
```

```
    int Left = node * 2;
    int Right = (node * 2) + 1;
    int mid = (b + e) / 2;
    update(Left, b, mid, i, j, x);
    update(Right, mid + 1, e, i, j, x);
    tree[node].sum = tree[Left].sum + tree[Right].sum + (e - b + 1) * tree[node].prop;
    //উপরে উঠার সময় পথের নোডগুলো আপডেট হবে
    //বাম আর ডান পাশের সাম ছাড়াও যোগ হবে নিচে অতিরিক্ত যোগ হওয়া মান
}


int query(int node, int b, int e, int i, int j, int carry = 0)
{
    if (i > e || j < b)
        return 0;

    if (b >= i and e <= j)
        return tree[node].sum + carry * (e - b + 1); //সাম এর সাথে যোগ হবে সেই রেঞ্জের সাথে অতিরিক্ত যত যোগ করতে বলেছে সেটা

    int Left = node << 1;
    int Right = (node << 1) + 1;
    int mid = (b + e) >> 1;

    int p1 = query(Left, b, mid, i, j, carry + tree[node].prop); //প্রপাগেট ভ্যালু বয়ে নিয়ে যাচ্ছে carry ভ্যারিয়েবল
    int p2 = query(Right, mid + 1, e, i, j, carry + tree[node].prop);

    return p1 + p2;
}
```

সমান্তর ধারা

১. একটি সমান্তর ধারার প্রথম পদ a এবং সাধারন অন্তর d হলে,
r -তম পদ $=a+(r-1)d$

২. প্রথম n সংখ্যক স্বাভাবিক সংখ্যার সমষ্টি $=\frac{n(n+1)}{2}$.
অর্থাৎ, $1+2+3+\ldots\ldots\ldots\ldots+n=\frac{n(n+1)}{2}$

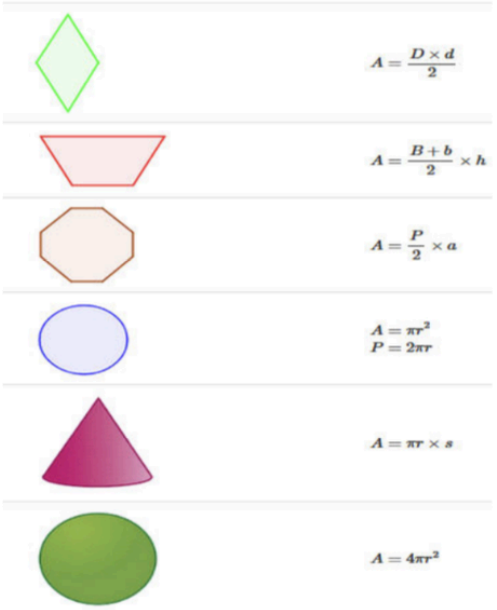
৩. প্রথম n সংখ্যক স্বাভাবিক সংখ্যার বর্গের সমষ্টি $=\frac{n(n+1)(2n+1)}{6}$
অর্থাৎ, $1^2+2^2+3^2+\ldots\ldots\ldots\ldots+n^2=\frac{n(n+1)(2n+1)}{6}$

৪. প্রথম n সংখ্যক স্বাভাবিক সংখ্যার ঘনের সমষ্টি $=\frac{n^2(n+1)^2}{4}$.
অর্থাৎ, $1^3+2^3+3^3+\ldots\ldots\ldots\ldots+n^3=\frac{n^2(n+1)^2}{4}$

৫. গুনোত্তর/সমানুপাতিক ধারার n তম পদ,
t_n =[প্রথমপদ × (সাধারন অনুপাত)$^{n-1}$]= ar^{n-1} এবং ইহার n সংখ্যক পদের যোগফল, $S_n=\frac{a(r^n-1)}{r-1}$ যখন $r>1$
আবার, $S_n=\frac{a(1-r^n)}{1-r}$, যখন $r<1$

$$\sum_{k=1}^{n} k=\frac{n(n+1)}{2}$$
$$\sum_{k=1}^{n} k^2=\frac{n(n+1)(2n+1)}{6}$$
$$\sum_{k=1}^{n} k^3=\frac{n^2(n+1)^2}{4}$$
$$\sum_{k=1}^{n} k(k+1)=\frac{n(n+1)(n+2)}{3}$$
$$\sum_{k=1}^{n} \frac{1}{k(k+1)}=\frac{n}{n+1}$$
$$\sum_{k=1}^{n} k(k+1)(k+2)=\frac{n(n+1)(n+2)(n+3)}{4}$$
$$\sum_{k=1}^{n} \frac{1}{k(k+1)(k+2)}=\frac{n(n+3)}{4(n+1)(n+2)}$$
$$\sum_{k=1}^{n} (2k-1)=n^2$$

Area



Sum of Squares = n(n+1)(2n+1)/6
Sum of n odd numbers = n^2
Sum of n even numbers = n(n+1)
Sum of cubes S = [n^2 (n + 1)^2]/4

$$S_n = \frac{a(1 - r^n)}{(1 - r)}$$

WHITEBOARD MATHS

FORMULA FOR THE SUM OF

$$S_n = \frac{n}{2}(2a + (n-1)d)$$

AN ARITHMETIC SERIES

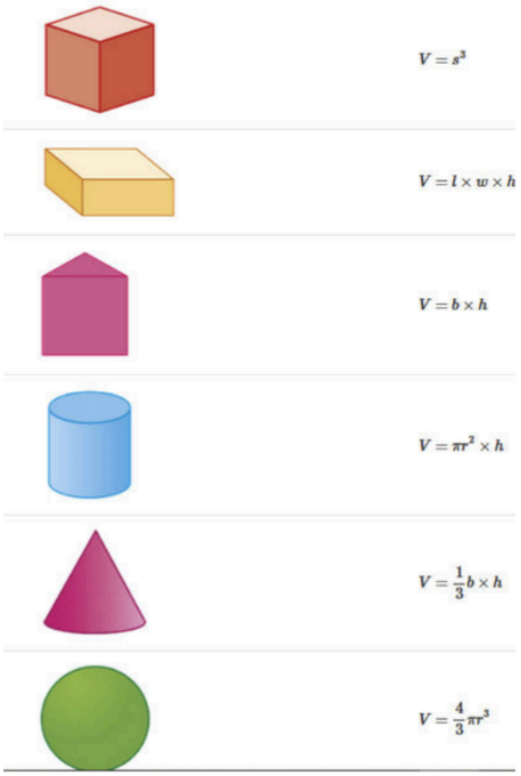
//NOD

$$(e_1 + 1) \cdot (e_2 + 1).$$

//SOD

$$\sigma(n) = \frac{p_1^{e_1+1} - 1}{p_1 - 1} \cdot \frac{p_2^{e_2+1} - 1}{p_2 - 1} \cdots \frac{p_k^{e_k+1} - 1}{p_k - 1}$$

Volume



<***** Geometry **********>**

Sum of n odd numbers = n^2

Sum of n even numbers = n(n+1)

Sum of geometric seriesNOD:

$$(e_1 + 1) \cdot (e_2 + 1).$$

SOD:

$$\sigma(n) = \frac{p_1^{e_1+1} - 1}{p_1 - 1} \cdot \frac{p_2^{e_2+1} - 1}{p_2 - 1} \cdots \frac{p_k^{e_k+1} - 1}{p_k - 1}$$

2.5 Derivatives/Integrals

$$\frac{d}{dx}\arcsin x = \frac{1}{\sqrt{1-x^2}} \qquad \frac{d}{dx}\arccos x = -\frac{1}{\sqrt{1-x^2}}$$

$$\frac{d}{dx}\tan x = 1 + \tan^2 x \qquad \frac{d}{dx}\arctan x = \frac{1}{1+x^2}$$

$$\int \tan ax = -\frac{\ln|\cos ax|}{a} \qquad \int x\sin ax = \frac{\sin ax - ax\cos ax}{a^2}$$

$$\int e^{-x^2} = \frac{\sqrt{\pi}}{2}\operatorname{erf}(x) \qquad \int xe^{ax}dx = \frac{e^{ax}}{a^2}(ax-1)$$

Integration by parts:

$$\int_a^b f(x)g(x)dx = [F(x)g(x)]_a^b - \int_a^b F(x)g'(x)dx$$

$$(V+W)\tan(v-w)/2 = (V-W)\tan(v+w)/2$$

where V, W are lengths of sides opposite angles v, w.

$$a\cos x + b\sin x = r\cos(x-\phi)$$
$$a\sin x + b\cos x = r\sin(x+\phi)$$

where $r = \sqrt{a^2+b^2}, \phi = \operatorname{atan2}(b,a)$.

2

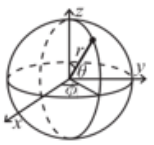
2.4.2 Quadrilaterals

With side lengths a, b, c, d, diagonals e, f, diagonals angle θ, area A and magic flux $F = b^2 + d^2 - a^2 - c^2$:

$$4A = 2ef \cdot \sin\theta = F\tan\theta = \sqrt{4e^2f^2 - F^2}$$

For cyclic quadrilaterals the sum of opposite angles is $180°$, $ef = ac + bd$, and $A = \sqrt{(p-a)(p-b)(p-c)(p-d)}$.

2.4.3 Spherical coordinates



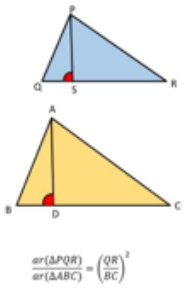
$$x = r\sin\theta\cos\phi \qquad r = \sqrt{x^2+y^2+z^2}$$
$$y = r\sin\theta\sin\phi \qquad \theta = \operatorname{acos}(z/\sqrt{x^2+y^2+z^2})$$
$$z = r\cos\theta \qquad \phi = \operatorname{atan2}(y,x)$$

WHITEBOARD MATHS

FORMULA FOR THE SUM OF

$$S_n = \frac{n}{2}(2a + (n-1)d)$$

AN ARITHMETIC SERIES



PROOF

STATEMENTS	REASONS
1. $\frac{ar(\Delta PQR)}{ar(\Delta ABC)} = \frac{1/2 \times QR \times PS}{1/2 \times BC \times AD}$	By Triangle Area Formula.
2. $\frac{ar(\Delta PQR)}{ar(\Delta ABC)} = \frac{QR \times PS}{BC \times AD}$	Cancel common Factor
3. $\frac{ar(\Delta PQR)}{ar(\Delta ABC)} = \frac{QR}{BC} \times \frac{PS}{AD}$	Equivalent Expression
4. $\frac{QR}{BC} = \frac{PS}{AD}$	The ratio of corresponding sides of similar triangles is equal to the ratio of the altitudes of the two triangles.

$$\frac{ar(\Delta PQR)}{ar(\Delta ABC)} = \left(\frac{QR}{BC}\right)^2$$

//Game Theory grandy number = mex{ for all reachables x from this state, grandy(x) }

// Pick's Theorem

Area of a polygon, A = i + (b / 2) - 1

Where, i = number of points inside the polygon, b = number of points on the boundary

2.7 Series

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots, (-\infty < x < \infty)$$

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots, (-1 < x \leq 1)$$

$$\sqrt{1+x} = 1 + \frac{x}{2} - \frac{x^2}{8} + \frac{2x^3}{32} - \frac{5x^4}{128} + \ldots, (-1 \leq x \leq 1)$$

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots, (-\infty < x < \infty)$$

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots, (-\infty < x < \infty)$$

2.4 Geometry

2.4.1 Triangles

Side lengths: a, b, c

Semiperimeter: $p = \frac{a+b+c}{2}$

Area: $A = \sqrt{p(p-a)(p-b)(p-c)}$

Circumradius: $R = \frac{abc}{4A}$

Inradius: $r = \frac{A}{p}$

Length of median (divides triangle into two equal-area triangles): $m_a = \frac{1}{2}\sqrt{2b^2 + 2c^2 - a^2}$

Length of bisector (divides angles in two):

$$s_a = \sqrt{bc\left[1 - \left(\frac{a}{b+c}\right)^2\right]}$$

Law of sines: $\frac{\sin\alpha}{a} = \frac{\sin\beta}{b} = \frac{\sin\gamma}{c} = \frac{1}{2R}$

Law of cosines: $a^2 = b^2 + c^2 - 2bc\cos\alpha$

Law of tangents: $\frac{a+b}{a-b} = \frac{\tan\frac{\alpha+\beta}{2}}{\tan\frac{\alpha-\beta}{2}}$

- Area of a Circle = A = $\pi \times r^2$
- Circumference of a Circle = $2\pi r$
- The curved surface area of a Cylinder = $2\pi rh$
- Total surface area of a Cylinder = $2\pi r(r + h)$
- Volume of a Cylinder = V = $\pi r^2 h$
- The curved surface area of a cone = πrl
- Total surface area of a cone = $\pi r(r+l) = \pi r[r+\sqrt{(h^2+r^2)}]$
- Volume of a Cone = V = $\frac{1}{3} \times \pi r^2 h$
- Surface Area of a Sphere = S = $4\pi r^2$
- Volume of a Sphere = V = $4/3 \times \pi r^3$

4.Parallelogram	Perimeter, P = 2(a + b) Area, A = bh Height, h = A/b Base, b = A/h Where, a and b are the sides of a parallelogram h = height of a parallelogram
5. Trapezium	Area, A = ½(a + b)h Height, h = 2A/(a + b) Base, b = 2(A/h) – a Where, a and b are the parallel sides h = distance between two parallel sides

8. Arc	Arc Length, L = rθ Area, A = ½r²θ Here, θ is the central angle is radians. Where, r = radius
9. Cube	Area, A = 6a² Volume, V = a³ Edge, a = V^⅓ Space diagonal = a√3 Where, a = side of a cube

2.2 Recurrences

If $a_n = c_1 a_{n-1} + \cdots + c_k a_{n-k}$, and $r_1, \ldots, r_k$ are distinct roots of $x^k + c_1 x^{k-1} + \cdots + c_k$, there are $d_1, \ldots, d_k$ s.t.

$$a_n = d_1 r_1^n + \cdots + d_k r_k^n.$$

Non-distinct roots r become polynomial factors, e.g. $a_n = (d_1 n + d_2) r^n$.

2.3 Trigonometry

$$\sin(v+w) = \sin v \cos w + \cos v \sin w$$
$$\cos(v+w) = \cos v \cos w - \sin v \sin w$$

$$\tan(v+w) = \frac{\tan v + \tan w}{1 - \tan v \tan w}$$
$$\sin v + \sin w = 2 \sin \frac{v+w}{2} \cos \frac{v-w}{2}$$
$$\cos v + \cos w = 2 \cos \frac{v+w}{2} \cos \frac{v-w}{2}$$

Linux——————————

```
{
"cmd" : ["g++ -std=c++14 $file_name -o $file_base_name && timeout 4s ./$file_base_name<inputf.in>outputf.in"],
"selector" : "source.c",
"shell": true,
"working_dir" : "$file_path"
}
```

Windows——————————

```
{
"cmd": ["g++.exe","-std=c++14", "${file}", "-o", "${file_base_name}.exe", "&&" , "${file_base_name}.exe<inputf.in>outputf.in"],
"selector":"source.cpp",
"shell":true,
"working_dir":"$file_path"
}
```