**Check if OJ/OS supports:**

1. **Your Blank Template**
2. **PDBS**
3. **Your bld tst**



***Combinatorics***

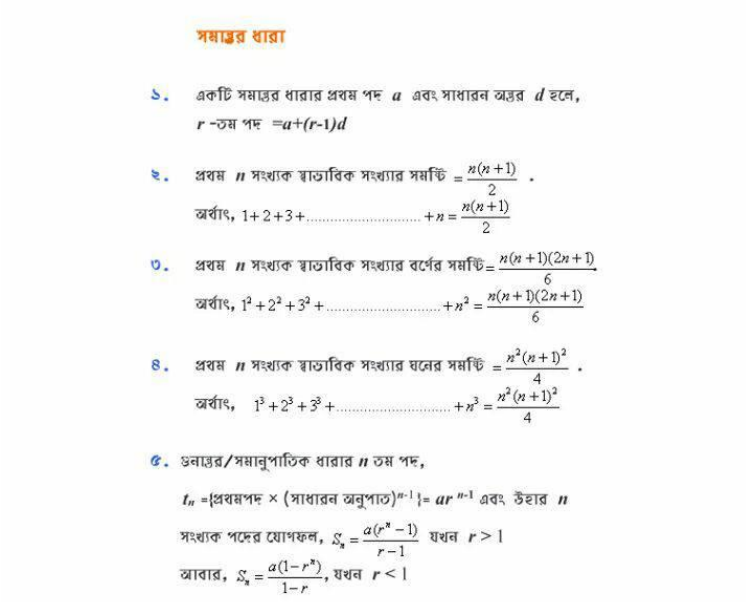
**nCr**

```
int mult(int a, int b) {
      return (1ll * a * b) % MOD;
}
int fact[MX], inv[MX], invfact[MX];
void init_INV() {
    fact[0] = invfact[0] = fact[1] = invfact[1] =
inv[1] = 1;
    for (int i = 2; i < MX; i++) {
        fact[i] = mult(fact[i - 1], i);
        inv[i] = mult(inv[MOD % i], MOD - MOD /
i);
        invfact[i] = mult(invfact[i - 1],
inv[i]);
    }
}
int ncr(int n, int r) {
    if (r > n) return 0;
    return (1LL * fact[n] * invfact[n - r] % MOD)
* 1LL * invfact[r] % MOD;
}
```

```
long long Derange(int n){

      if(n==1) return 0;
      if(n==2 || n==0) return 1;
      long long ret = dpDerange[n];
      if(ret!=-1) return ret;

ret = ((n-1)%mod *
(Derange(n-1)%mod+Derange(n-2)%mod)%mod)%mod;

    return ret;

}
```

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





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, $1+2+3+\ldots\ldots+n = \frac{n(n+1)}{2}$

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

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

৫. গুনোত্তর/সমানুপাতিক ধারার $n$ তম পদ,
$t_n$ =(প্রথমপদ × (সাধারন অনুপাত)$^{n-1}$}= $ar^{n-1}$ এবং উহার $n$ 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 $r<1$

***//Stars and bars***

*The number of ways to put identical objects into labelled boxes is*

$$\binom{n+k-1}{n}.$$

***//inclusion-exclusion***

*Add if odd and Subtract if even*

```
loop(i, 1, choto) {
        ll tmp = nCr(choto, i) * nPr(m - i, n - i) % M;
        if(i & 1) extra = (extra + tmp) % M;
        else extra = ((extra - tmp) % M + M) % M;
}
```

***Number theory***

**Matrix Exponentiation**

```
//Multiply two square matrices
vector<vector<int>>multiply(vector<vector<int>>&a, vector<vector<int>>&b) {
    int sz = a.size();
    vector<vector<int>>ans(sz, vector<int>(sz, 0));
    for (int i = 0; i < sz; i++) {
        for (int j = 0; j < sz; j++) {
            for (int k = 0; k < sz; k++) {
                ans[i][j] += (a[i][k] * b[k][j]);
            }
        }
    }
    return ans;
}
vector<vector<int>>matrix(vector<vector<int>>&a, int n) {
    int sz = a.size();
    if (n == 0) {
        vector<vector<int>>ans(sz, vector<int>(sz, 0));
        for (int i = 0; i < sz; i++) {
            ans[i][i] = 1;
            return ans;
        }
    }
    else if (n == 1) {
        return a;
    }
    vector<vector<int>> temp = matrix(a, n / 2);
    vector<vector<int>>ans = multiply(temp, temp);
    if ((n & 1)) {
        ans = multiply(ans, a);
    }
    return ans;
}
//calculate (matrix)^n
void matrixExponentiation(vector<vector<int>>&a, int n) { // we need to pass a matrix here
    // vector<vector<int>>a = {{1,1},{1,0}};
    vector<vector<int>>ans =  matrix(a, n);
    cout << ans[0][1] << endl;
}
```

**//Segmented sieve**

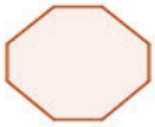
```
vector<ll>primes;
void sieve(ll mxN){
    vector<char>isprime(mxN+1,true);
    primes.pb(2);
    for (ll i = 3; i <= mxN; i += 2) {
        if (isprime[i]) {
            primes.pb(i);
            for (ll j = i*i; j <= mxN; j += i)
                isprime[j] = false;
        }
    }
}
```

```
void seg_sieve(ll l, ll ss){
    vector<char>block(ss, true);
    for (auto p : primes) {
        ll x = (l + p - 1) / p;
        ll j = (x*p) - l;
        for ( ; j < ss; j += p)
            block[j] = false;
    }
    for (int i = 0; i < ss; i++) {

    }
}
```

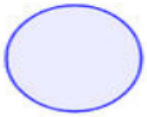
**Area:**

$$A = \frac{D \times d}{2}$$

$$A = \frac{B + b}{2} \times h$$



$$A = \frac{P}{2} \times a$$



$$A = \pi r^2$$
$$P = 2\pi r$$

$$A = \pi r \times s$$

$$A = 4\pi r^2$$

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

**Volume:**

$$V = s^3$$

$$V = l \times w \times h$$

$$V = b \times h$$

$$V = \pi r^2 \times h$$

$$V = \frac{1}{3} b \times h$$

$$V = \frac{4}{3}\pi r^3$$

**//NOD**

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

**//SOD**

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

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

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

**//phi**

```
vector<ll> phi(N);

void calculatephi() {
        for(ll i = 1; i < N; i++) {
                phi[i] = i;
        }
        for(ll i = 2; i < N; i++) {
                if(phi[i] == i) {
                        for(ll j = 2 * i; j < N;
j += i) {
                                phi[j] *= (i -
1);
                                phi[j] /= i;
                        }
                        phi[i] = i - 1;
                }
        }
}
```

**//Pollard Rho**

```
namespace PollardRho {
  mt19937
rnd(chrono::steady_clock::now().time_since_epoch(
).count());
  const int P = 1e6 + 9;
  ll seq[P];
  int primes[P], spf[P];
  inline ll add_mod(ll x, ll y, ll m) {
    return (x += y) < m ? x : x - m;
  }
  inline ll mul_mod(ll x, ll y, ll m) {
    ll res = __int128(x) * y % m;
    return res;
    // ll res = x * y - (ll)((long double)x * y /
m + 0.5) * m;
    // return res < 0 ? res + m : res;
  }
  inline ll pow_mod(ll x, ll n, ll m) {
    ll res = 1 % m;
    for (; n; n >>= 1) {
      if (n & 1) res = mul_mod(res, x, m);
      x = mul_mod(x, x, m);
    }
    return res;
  }
  // O(it * (logn)^3), it = number of rounds
performed
  inline bool miller_rabin(ll n) {
    if (n <= 2 || (n & 1 ^ 1)) return (n == 2);
    if (n < P) return spf[n] == n;
    ll c, d, s = 0, r = n - 1;
    for (; !(r & 1); r >>= 1, s++) {}
    // each iteration is a round
    for (int i = 0; primes[i] < n && primes[i] <
32; i++) {
      c = pow_mod(primes[i], r, n);
      for (int j = 0; j < s; j++) {
        d = mul_mod(c, c, n);
        if (d == 1 && c != 1 && c != (n - 1))
return false;
        c = d;
      }
      if (c != 1) return false;
    }
    return true;
  }
  void init() {
    int cnt = 0;
    for (int i = 2; i < P; i++) {
      if (!spf[i]) primes[cnt++] = spf[i] = i;
      for (int j = 0, k; (k = i * primes[j]) < P;
j++) {
        spf[k] = primes[j];
        if (spf[i] == spf[k]) break;
      }
    }
  }
  // returns O(n^(1/4))
  ll pollard_rho(ll n) {
    while (1) {
      ll x = rnd() % n, y = x, c = rnd() % n, u =
1, v, t = 0;
      ll *px = seq, *py = seq;
      while (1) {
        *py++ = y = add_mod(mul_mod(y, y, n), c,
n);
        *py++ = y = add_mod(mul_mod(y, y, n), c,
n);
        if ((x = *px++) == y) break;
        v = u;
        u = mul_mod(u, abs(y - x), n);
        if (!u) return __gcd(v, n);
        if (++t == 32) {
          t = 0;
          if ((u = __gcd(u, n)) > 1 && u < n)
return u;
        }
      }
      if (t && (u = __gcd(u, n)) > 1 && u < n)
return u;
    }
  }
  vector<ll> factorize(ll n) {
    if (n == 1) return vector <ll>();
    if (miller_rabin(n)) return vector<ll> {n};
    vector <ll> v, w;
    while (n > 1 && n < P) {
      v.push_back(spf[n]);
      n /= spf[n];
    }
    if (n >= P) {
      ll x = pollard_rho(n);
      v = factorize(x);
      w = factorize(n / x);
      v.insert(v.end(), w.begin(), w.end());
    }
    return v;
  }
//   PollardRho::init();
```

```
//   vector<int> ax =
PollardRho::factorize(1000000007);
//   print(ax);
}

//Milar rabin
using u64 = uint64_t;
using u128 = __uint128_t;

u64 binpower(u64 base, u64 e, u64 mod) {
    u64 result = 1;
    base %= mod;
    while (e) {
        if (e & 1)
            result = (u128)result * base % mod;
        base = (u128)base * base % mod;
        e >>= 1;
    }
    return result;
}

bool check_composite(u64 n, u64 a, u64 d, int s)
{
    u64 x = binpower(a, d, n);
    if (x == 1 || x == n - 1)
        return false;
    for (int r = 1; r < s; r++) {
        x = (u128)x * x % n;
        if (x == n - 1)
            return false;
    }
    return true;
};

bool MillerRabin(u64 n) { // returns true if n is
prime, else returns false.
    if (n < 2)
        return false;

    int r = 0;
    u64 d = n - 1;
    while ((d & 1) == 0) {
        d >>= 1;
        r++;
    }

    for (int a : {2, 3, 5, 7, 11, 13, 17, 19, 23,
29, 31, 37}) {
        if (n == a)
            return true;
        if (check_composite(n, a, d, r))
            return false;
    }
    return true;
}
```

```
//extended euclid
ll extended_euclid(ll a, ll b, ll &x, ll &y) {
  if (b == 0) {
    x = 1; y = 0;
    return a;
  }
  ll x1, y1;
  ll d = extended_euclid(b, a % b, x1, y1);
  x = y1;
  y = x1 - y1 * (a / b);
  return d;
}
ll inverse(ll a, ll m) {
  ll x, y;
  ll g = extended_euclid(a, m, x, y);
  if (g != 1) return -1;
  return (x % m + m) % m;
}
//Game Theory
grandy number = mex{ for all reachables x from
this state, grandy(x) }
My Grandy Mex:
void GrandyMEX(int n){
    memset(vis , false , sizeof(vis));

    if(n < 2){
        grand[n] = 0;
        return;
    }
    if(grand[n] != -1) return;

    for(int i = 1 ; i*2 <= n; i++){
        vis[grand[n-i]] = true;
    }

    for(int i = 0 ; i <= mx ; i++){
        if(!vis[i]){
            grand[n] = i;
            return;
        }
    }
}
//inside main
for(int i = 0 ; i <= mx ; i++){
        GrandyMEX(i);
        // cout<<i<<" "<<grand[i]<<endl;
}
//Graph



// dijksta's algorithm
const int mx;

struct graph{
    int node;
    int cost;
};
```

```
vector<graph>g[mx];
bool vis[mx];
int dis[mx];

class cmp{
    public:
        bool operator() (graph &A, graph &B) {
            if (A.cost > B.cost) return true;
            return false;
        }
};

void dijkstra(int source){
    priority_queue<graph, vector<graph>, cmp>pq;
    pq.push({source,0});
    while (!pq.empty()) {
        graph cur = pq.top();
        pq.pop();
        int node = cur.node;
        int cost = cur.cost;
        if (vis[node]) continue;
        dis[node] = cost;
        vis[node] = true;
        for(int i=0;i<g[node].size();i++) {
            int nxt = g[node][i].node;
            int nxtc = g[node][i].cost;
            if (!vis[nxt]) {
                pq.push({nxt,nxtc+cost});
            }
        }
    }
}
```

**//Krushkal MST**

```
vector<ll> par(N);
vector<array<ll,3>> edge;
ll n, m;

ll find(ll x) {
        return par[x] = (par[x] == x ? x :
find(par[x]));
}

ll kruskal() {
        sort(all(edge));
        for(int i = 1; i <= n; i++) par[i] = i;
        ll cost = 0, cnt = 0;
        for(auto [w, x, y] : edge) {
                x = find(x);
                y = find(y);
                if(x != y) {
                        par[x] = y;
                        cost += w;
                        cnt++;
                }
        }
        return (cnt == n-1 ? cost : -1);
}
```

**//Prims MST**

```
struct prim
{
    const static int N = 2e6 + 10; //set N here
    vector<ll>vis;
    ll cost = 0;
    void init(ll st, ll n = N){ //starting node
and size
        vis.resize(n+5);
        priority_queue< pll, vector<pll> ,
greater<pll>> list;
        list.push({0, st});
        while(!list.empty()){
            pll now = list.top();
            vis[now.s] = 1;
            list.pop();
            //i am using var "now" , change here
            cost += now.f;
            for(auto a: adj[now.s]){
                if(!vis[a.f]){
                    list.push(mp(a.s, a.f));
                }
            }
            while(!list.empty() &&
vis[list.top().s]) {
                list.pop();
            }
        }
    }
};
```

**//Cycle check (directed)**

```
bool isCyclic(int N, vector<int> adj[]) {
    queue<int> q;
    vector<int> indegree(N, 0);
    for(int i = 0;i<N;i++) {
        for(auto it: adj[i]) {
            indegree[it]++;
        }
    }

    for(int i = 0;i<N;i++) {
        if(indegree[i] == 0) {
            q.push(i);
        }
    }
    int cnt = 0;
    while(!q.empty()) {
        int node = q.front();
        q.pop();
        cnt++;
        for(auto it : adj[node]) {
            indegree[it]--;
            if(indegree[it] == 0) {
                q.push(it);
            }
        }
    }
    if(cnt == N) return false;
    return true;
}
```

**LCA**

```
const int mx = 1e5 + 5;
const int k = 18;
vector<int>g[mx];
int parrent[mx][k];
vector<int>height(mx);

void dfs(int node, int h){
    height[node] = h;
    for (auto it : g[node]) {
        if (!height[it]) {
            dfs(it,h+1);
            parrent[it][0] = node;
        }
    }
}

void buildSparseTable(int n){
    for (int j = 1; j < k; j++) for (int i = 1; i
<= n; i++)
        if (parrent[i][j-1] != -1) {
            parrent[i][j] =
parrent[parrent[i][j-1]][j-1];
            //ma[i][j] = max(ma[i][j-1],
ma[parrent[i][j-1]][j-1]);
        }
}

int LCA(int u, int v){
    if (height[u] > height[v]) swap(u,v);
    int dif = height[v] - height[u];
    for (int i = k-1; i >= 0; i--)
        if (dif&(1<<i)) {
            v = parrent[v][i];
        }
    if (u == v) return u;
    for (int i = k-1; i >= 0; i--) {
        if (parrent[u][i] != parrent[v][i]) {
            u = parrent[u][i];
            v = parrent[v][i];
        }
    }
    return parrent[u][0];
}
```

**AP->**

```
const int mx = 1e4 + 5;
vector<int>g[mx];
vector<int>Time(mx),Low(mx),check(mx);

void dfs(int node, int pa, int &t){
    Time[node] = Low[node] = t++;
    int z = 0;
    for (auto it : g[node]) {
        if (it == pa) continue;
        if (!Time[it]) {
            dfs(it,node,t); z++;
            Low[node] = min(Low[node],Low[it]);
            if (Low[it] >= Time[node] && pa !=
-1)
                check[node] = 1;
        } else {
            Low[node] = min(Low[node],Time[it]);
        }
    }
    if (pa == -1 && z > 1)
        check[node] = 1;
}
```

**//bipartite graph**

```
void bfs(int n) {
    queue<int> q;
    color[n] = 1;
    q.push(n);
    while(q.size()) {
        int x = q.front();
        q.pop();
        for(auto it : g[x]) {
            if(!color[it]) {
                color[it] = color[x] % 2 + 1;
                q.push(it);
            } else if(color[it] == color[x]) {
                cout << IMPOSSIBLE << endl;
                exit(0);
            }
        }
    }while(q.size())
}
```

**// bellman_ford algorithm**

```
vector<pair<ll,ll>> g[N];
vector<ll> dist(N), par(N);
ll n, m;

void bellman_ford(ll src) {
        for(ll i = 0; i <= n; i++) dist[i] = INF;
        dist[src] = 0;
        for(ll step = 0; step < n - 1; step++) {
                //1 based indexing
                for(ll i = 1; i <= n; i++) {
                        //it.ff = node, it.ss =
weight
                        for(auto it : g[i]) {
                                if(dist[i] +
it.ss < dist[it.ff]) {

par[it.ff] = i;

dist[it.ff] = dist[i] + it.ss;
                                }
                        }
                }
        }
}
```

```
void cycle_detect() {
        ll cycle = 0;
        for(ll i = 1; i <= n; i++) {
                for(auto it : g[i]) {
                        if(dist[i] + it.ss <
dist[it.ff]) {
                                cycle = it.ff;
                                break;
                        }
                }
        }
        if(cycle == 0) {
                cout << "NO" << endl;
        } else {
                cout << "YES" << endl;
                for(ll i = 0; i < n; i++) cycle =
par[cycle];
                vector<ll> ans;
                ans.pb(cycle);
                for(ll i = par[cycle]; i !=
cycle; i = par[i]) {
                        ans.pb(i);
                }
                ans.pb(cycle);
                reverse(all(ans));
                for(auto it : ans) cout << it <<
" ";
                cout << endl;
        }
}
```

**// floyd_warshal algorithm**

```
vector<vector<ll>> dist(N, vector<ll>(N));
ll n, m, q;

void floyd_warshaal() {
        for(int k = 1; k <= n; k++) {
                for(int i = 1; i <= n; i++) {
                        for(int j = 1; j <= n;
j++) {
                                dist[i][j] =
min(dist[i][j], dist[i][k] + dist[k][j]);
                        }
                }
        }
}

for(int i = 1; i <= n; i++) {
        for(int j = 1; j <= n; j++) {
                if(i == j) dist[i][j] = 0;
                else dist[i][j] = INF;
        }
}
```

**//DP**

**//DP LIS**

```
int lis(int arr[], int n)
{
    int lis[n];

    lis[0] = 1;

    /* Compute optimised LIS values in
       bottom up manner */
    for (int i = 1; i < n; i++) {
        lis[i] = 1;
        for (int j = 0; j < i; j++)
            if (arr[i] > arr[j] && lis[i] <
lis[j] + 1)
                lis[i] = lis[j] + 1;
    }

    // Return maximum value in lis[]
    return *max_element(lis, lis + n);
}
```

**//lcs**

```
string s, t;

ll lcs() {
        ll n = s.size();
        ll m = t.size();
        vector<vector<ll>> dp(n + 1, vector<ll>(m
+ 1));
        for(ll i = 1; i <= n; i++) {
                for(ll j = 1; j <= m; j++) {
                        if(s[i - 1] == t[j - 1])
dp[i][j] = dp[i - 1][j - 1] + 1;
                        else dp[i][j] = max(dp[i
- 1][j], dp[i][j - 1]);
                }
        }
        return dp[n][m];
}
```

**//Maximum subarray sum**

```
vector<vector<ll>> a(n+1, vector<ll>(n+1));
for(int i = 1; i <= n; i++) {
        for(int j = 1; j <= n; j++) {
                cin >> a[i][j];
                a[i][j] += a[i][j-1];
        }
}

// if empty subarray allowed... then set ans = 0
, cur = 0 and use cur = max(0LL, cur + x);
ll ans = -INF;
for(int left = 1; left <= n; left++) {
        for(int right = left; right <= n;
right++) {
                ll cur = -INF;
                for(int i = 1; i <= n; i++) {
                        ll x = a[i][right] -
a[i][left-1];
                        cur = max(x, cur + x);
                        ans = max(ans, cur);
                }
        }
}
```

**//Segment tree**

Point update:

```
#define mx 100001
int arr[mx];
int tree[mx * 4];


void init(int node, int b, int e)
{
    if (b == e) {
        tree[node] = arr[b];
        return;
    }
    int Left = node * 2;
    int Right = node * 2 + 1;
    int mid = (b + e) / 2;
    init(Left, b, mid);
    init(Right, mid + 1, e);
    tree[node] = tree[Left] + tree[Right];
}


int query(int node, int b, int e, int i, int j)
{
    if (i > e || j < b)
        return 0;
    if (b >= i && e <= j)
        return tree[node];
    int Left = node * 2;
    int Right = node * 2 + 1;
    int mid = (b + e) / 2;
    int p1 = query(Left, b, mid, i, j);
    int p2 = query(Right, mid + 1, e, i, j);
    return p1 + p2
}
```

```
void update(int node, int b, int e, int i, int
newvalue)
{
    if (i > e || i < b)
        return;
    if (b >= i && e <= i) {
        tree[node] = newvalue;
        return;
    }
    int Left = node * 2;
    int Right = node * 2 + 1;
    int mid = (b + e) / 2;
    update(Left, b, mid, i, newvalue);
    update(Right, mid + 1, e, i, newvalue);
    tree[node] = tree[Left] + tree[Right];
}
```

RangeUpdate:

```
struct info {
    int prop, sum;
} tree[mx * 4];
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)
    {
        tree[node].sum += ((e - b + 1) * x);
        tree[node].prop += x;
        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;
}
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);
    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);
    int p2 =
query(Right,mid+1,e,i,j,carry+tree[node].prop);
    return p1 + p2;
```

```
}
```

2D segment tree:

```
// initiate
void initY(ll nodeX , ll startX , ll endX , ll
nodeY , ll startY , ll endY){
    if(startY == endY){
        if(startX == endX){
            tree[nodeX][nodeY] =
arr[startX][startY];
        }else{
            tree[nodeX][nodeY] = tree[nodeX *
2][nodeY] + tree[(nodeX * 2) + 1][nodeY];
        }
        return;
    }
    ll left = nodeY * 2;
    ll right = left + 1;
    ll mid = (startY + endY) / 2;
    initY(nodeX , startX , endX , left , startY ,
mid);
    initY(nodeX , startX , endX , right , mid + 1
, endY);
    tree[nodeX][nodeY] = tree[nodeX][left] +
tree[nodeX][right];
}
void initX(ll nodeX , ll startX , ll endX){
    if(startX == endX){
        initY(nodeX , startX , endX , 1 , 1 , n);
        return;
    }
    ll left = nodeX * 2;
    ll right = left + 1;
    ll mid = (startX+endX)/2;
    initX(left , startX , mid);
    initX(right , mid + 1 , endX);
    initY(nodeX , startX , endX , 1 , 1 , n);
}
//query
ll queryY(ll nodeX , ll nodeY , ll startY , ll
endY , ll y1 , ll y2){
    if(y1 > endY || y2 < startY)
        return 0;
    if(startY == endY){
        return tree[nodeX][nodeY];
    }
    if(y1 <= startY && y2 >= endY){
        return tree[nodeX][nodeY];
    }
    ll left = nodeY * 2;
    ll right = left + 1;
    ll mid = (startY + endY) / 2;
    ll q1 = queryY(nodeX , left , startY , mid ,
y1 , y2);
    ll q2 = queryY(nodeX , right , mid + 1 , endY
, y1 , y2);
    return q1 + q2;
}
ll queryX(ll nodeX , ll startX , ll endX , ll x1
, ll x2 , ll y1 , ll y2){
    if(x1 > endX || x2 < startX)    return 0;

    if(startX == endX)
        return queryY(nodeX , 1 , 1 , n , y1 ,
y2);
    if(x1 <= startX && x2 >= endX)
        return queryY(nodeX , 1 , 1 , n , y1 ,
y2);

    ll left = nodeX * 2;
    ll right = left + 1;
    ll mid = (startX + endX) / 2;
    ll q1 = queryX(left , startX , mid , x1 , x2
, y1 , y2);
    ll q2 = queryX(right , mid + 1 , endX , x1 ,
x2 , y1 , y2);
    return q1 + q2;
}
```

**ordered Set**

```
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
//forNums
greater.
typedef tree<ll, null_type, less<ll>,
rb_tree_tag, tree_order_statistics_node_update>
pbd;
//forPair
typedef tree<pair<ll , ll>, null_type,
less<pair<ll , ll>>, rb_tree_tag,
tree_order_statistics_node_update> pbds;

set.find_by_order(n); //finding nth element
set.order_of_key(n); //number of elment before n;
set.upper_bound(n); // upper bound of n;
set.lower_bound(n); // lower bound of n;
set.erase(n) // erase n;
```

**//fenwick tree**

```
const int N = 1e5 + 10;
int n;
int arr[N];

void update(int ind, int val) {
    while (ind <= n) {
        arr[ind] += val;
        ind += (ind & -ind);
    }
}

int query(int ind) {
    int sum = 0;
    while (ind > 0) {
        sum += arr[ind];
        ind -= (ind & -ind);
    }
    return sum;
}
```

```
//Next Greater, next lower
// next_greater_left
stack<ll> stlmax;
vector<ll> lmax(n, -1);
for(i = n-1; i >= 0; i--) {
    while(!stlmax.empty() && a[stlmax.top()] <
a[i]) {
        lmax[stlmax.top()] = i;
        stlmax.pop();
    }
    stlmax.push(i);
}
next_greater_right
int nxtGreater[n + 1];
    stack<pair<int, int>> s;
    s.push({v[0], 0});
    for (int i = 1; i < n; i++)
    {

        if (s.empty()) {
            s.push({v[i], i});
            continue;
        }
        while (!s.empty() && s.top().F < v[i])
        {
            nxtGreater[s.top().S] = v[i];
            s.pop();
        }
        s.push({v[i], i});
    }
    while (!s.empty()) {
        nxtGreater[s.top().S] = -1;
        s.pop();
    }

// next_lower_left
stack<ll> stlmin;
vector<ll> lmin(n, -1); //0 based indexing
for(i = n-1; i >= 0; i--) {
    while(!stlmin.empty() && a[stlmin.top()] >
a[i]) {
        lmin[stlmin.top()] = i;
        stlmin.pop();
    }
    stlmin.push(i); //using index
}
// next_lower_right
stack<ll> strmin;
vector<ll> rmin(n, -1);
for(i = 0; i < n; i++) {
    while(!strmin.empty() && a[strmin.top()] >
a[i]) {
        rmin[strmin.top()] = i;
        strmin.pop();
    }
    strmin.push(i);
}
```

***String***

```
//KMP


vector<int> get_lps(string s){
    int n = s.size();
    vector<int>lps(n,0);
    int i = 1, j = 0;
    while (i < n) {
        if (s[i] == s[j]) {
            lps[i] = j+1;
            i++; j++;
        } else {
            if (j) j = lps[j-1];
            else i++;
        }
    }
    return lps;
}




vector<int> kmp(string s, string t){
    vector<int> lps = get_lps(t);
    int i = 0, j = 0;
    int n = s.size(), m = t.size();
    vector<int>ixs;
    while (i < n) {
        if (s[i] == t[j]) {
            i++; j++;
        } else {
            if (j) j = lps[j-1];
            else i++;
        }
        if (j == m) {
            ixs.pb(i-m+1);
            j = lps[j-1];
        }
    }
    return ixs;
}
```

**Trie**

```
typedef struct trie
{
      typedef struct node
      {
            node* nxt[2];
            int cnt = 0;

            node()
            {
                  nxt[0] = nxt[1] = NULL;
                  cnt = 0;
            }

      }Node;
```

```
	Node* head;

	trie() { head = new Node(); }

	void insert(int x)
	{
		Node* cur = head;
		for(int i = 30; i >= 0; i--)
		{
			int b = (x >> i) & 1;
			if(!cur -> nxt[b])
				cur -> nxt[b] = new
Node();
			cur = cur -> nxt[b];
			cur -> cnt++;
		}
	}

	void remove(int x)
	{
		Node* cur = head;
		for(int i = 30; i >= 0; i--)
		{
			int b = (x >> i) & 1;
			cur = cur -> nxt[b];
			cur -> cnt--;
		}
	}

	int maxxor(int x)
	{
		Node* cur = head;
		int ans = 0;
		for(int i = 30; i >= 0; i--)
		{
			int b = (x >> i) & 1;
			if(cur -> nxt[!b] && cur ->
nxt[!b] -> cnt > 0)
			{
				ans += (1LL << i);
				cur = cur -> nxt[!b];
			}
			else
				cur = cur -> nxt[b];
		}
		return ans;
	}
}Trie;
```

**Suffix array**

```
vector<int> suffixArray(string s){
    s.pb('$');
    int n = s.size();
    vector<int>sa(n), uc(n);
    {
        vector<ii>ta(n); // tmp array for sorting
        for (int i = 0; i < n; i++) ta[i] =
{s[i],i};
        sort(ta.begin(),ta.end());
        for (int i = 0; i < n; i++) sa[i] =
ta[i].second;
        uc[sa[0]] = 0;
        for (int i = 1; i < n; i++) {
            if (ta[i].first == ta[i-1].first)
uc[sa[i]] = uc[sa[i-1]];
            else uc[sa[i]] = uc[sa[i-1]] + 1;
        }
    }
    int k = 0;
    while ((1<<k) < n) {
        vector< pair<pair<int, int>, int>> ta(n);
        for (int i = 0; i < n; i++)
            ta[i] = {{uc[i], uc[(i + (1<<k)) %
n]}, i};
        sort(ta.begin(),ta.end());
        for (int i = 0; i < n; i++) sa[i] =
ta[i].second;
        uc[sa[0]] = 0;
        for (int i = 1; i < n; i++) {
            if (ta[i].first == ta[i-1].first)
uc[sa[i]] = uc[sa[i-1]];
            else uc[sa[i]] = uc[sa[i-1]] + 1;
        }
        k++;
    }
    return sa;
}
```

**//Aho-Corasick Algorithm**

```
//  template for AhoCorasic
const int INF = 1 << 30;
const int N = 1e5 + 7;
struct AhoCorasic {
  static const int32_t K = 26;
  struct node {
    bool ok = false;
    int next[26];
    int res = INF;
    int frq;
    vector<int> cntPos;
    node() {
      fill(begin(next), end(next), 0);
    }
  };
  int32_t a = 'a';
  // tri for storing string
  vector<node> t;
  // ind of the patarn
  vi ind;
  // suffix link for the node
```

```
  vi link;
  // count for the patarn
  vi cnt;
  int32_t s;
  AhoCorasic() {
    t.reserve(N);
    cnt.reserve(N);
    link.reserve(N);
    ind.reserve(N);
    t.emplace_back();
    s = 1;
  }

  vector<int> &insert(vector<string> &patarns, vi
&fre) {
    int n = patarns.size();
    ind.resize(n);
    for (int i = 0; i < n; i++) {
      // if(patarns[i].size()>limit) continue;
      ind[i] = insert(patarns[i]);
      t[ind[i]].frq = fre[i];
    }
    return ind;
  }

  int insert(string &pat) {
    int cn = 0;
    for (auto it : pat) {
      int c = it - a;
      if (!t[cn].next[c]) {
        t[cn].next[c] = s++;
        t.emplace_back();
      }
      cn = t[cn].next[c];
    }
    t[cn].ok = true;
    return cn;
  }

  void buildLink() {
    link.resize(s, 0);
    queue<int32_t> Q;

    for (int i : t[0].next) {
      if (i)
        Q.push(i);
    }

    while (!Q.empty()) {

      int par = Q.front();
      Q.pop();

      for (int i = 0; i < K; i++) {
        int child = t[par].next[i];
        if (!child) {
          continue;
        }

        int j = link[par];
        while (j && t[j].next[i] == 0) {
          j = link[j];
        }
        link[child] = t[j].next[i];
        Q.push(child);
      }
    }
  }

  vector<int> &search(string &text) {
    int n = text.size();

    int cn = 0;

    // cnt.resize(s, 0);
    int koi = 0;
    for (auto it : text) {
      int c = it - a;
      while (cn && t[cn].next[c] == 0) {
        cn = link[cn];
      }
      int temp = cn = t[cn].next[c];

      while (temp) {
        if (t[temp].ok)
          t[temp].cntPos.push_back(koi);

        if (t[temp].ok && t[temp].cntPos.size()
>= t[temp].frq) {
          t[temp].res = min(t[temp].res,
t[temp].cntPos.back() -
t[temp].cntPos[t[temp].cntPos.size() -
t[temp].frq]);
        }
        // cnt[temp]++;
        temp = link[temp];
      }
      koi++;
    }
    return cnt;
  }

  void makeAllTransition() {
    queue<int32_t> Q;
    Q.push(0);
    while (!Q.empty()) {

      int top = Q.front();
      Q.pop();
      // dbg(top);
      int ind = 0;
      for (int x : t[top].next) {
        // dbg(ind, x);
        if (x != 0)
          Q.push(x);
        else
          t[top].next[ind] =
t[link[top]].next[ind];
        ind++;
      }
    }
  }
```

```
  void clear() {
    t.clear();
    s = 0;
    cnt.clear();
    ind.clear();
    link.clear();
  }
};
```

```
//PBDS
// 1. delete a given number from the list
os.erase(5);
// when you use less equal
erase(ms.find_by_order(ms.order_of_key(value)))

// 2. find a given number(7)
if (os.find(7) == os.end())
    cout << "not found\n";
else
    auto idx = os.find(7);
// 3. how many numbers are smaller than a given
value(7)
cout << os.order_of_key(7);
// 4. delete the k'th smallest number
os.erase(os.find_by_order(k));
// 5. delete the smallest number from the list
os.erase(os.begin());
// 6. delete the greatest number from the list
os.erase(os.rbegin());
// 7. what is the smallest number which is
greater than or equal to a given number(7)
cout << *os.lower_bound(7) << "\n";
// 8. what is the smallest number which is
greater than to a given number(7)
cout << *os.upper_bound(7) << "\n";
```

**//Custom Hash**

```
struct custom_hash {
    static uint64_t splitmix64(uint64_t x) {
        x += 0x9e3779b97f4a7c15;
        x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
        x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
        return x ^ (x >> 31);
    }

    size_t operator()(uint64_t x) const {
        static const uint64_t FIXED_RANDOM =
chrono::steady_clock::now().time_since_epoch().co
unt();
        return splitmix64(x + FIXED_RANDOM);
    }
};
```

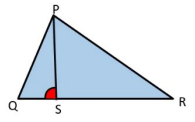
***//Geometry***

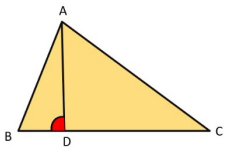
**Intersection check of two line**

```
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;
    if (val > 0) return 1;
    return 2;
}

bool check(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;
}
```

## Area of all kind of polygon (include triangle,rectangle etc):

```
//assign x[n]=x[0] and y[n]=y[0] and increase n
to n+1 before calling..
long double polygonArea(int x[],int y[],int n)
{
    int area = 0;
    for(int i=0;i<n;i++)
    {
        area += x[i]*y[i+1]-x[i+1]*y[i];
    }
    long double ar = area;
    ar/=2;
    return ar;
}
```




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

| 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. |

## Some important Information:

- 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 = $\pi rl$
- 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)<br>Area, A = bh<br>Height, h = A/b<br>Base, b = A/h<br>Where,<br>a and b are the sides of a parallelogram<br>h = height of a parallelogram |
| 5. Trapezium | Area, A = ½(a + b)h<br>Height, h = 2A/(a + b)<br>Base, b = 2(A/h) – a<br>Where,<br>a and b are the parallel sides<br>h = distance between two parallel sides |
| 8. Arc | Arc Length, L = rθ<br>Area, A = ½r²θ<br>Here, θ is the central angle is radians.<br>Where,<br>r = radius |
| 9. Cube | Area, A = $6a^2$<br>Volume, V = $a^3$<br>Edge, a = $V^{1/3}$<br>Space diagonal = $a\sqrt{3}$<br>Where,<br>a = side of a cube |

## Sourav:

```
ll ExtendedEucledeangcd(ll a, ll b, ll &x, ll &y)
{
    if (b == 0LL) {
        x = 1;
        y = 0;
        return a;
    }
    ll x1, y1;
    ll d = ExtendedEucledeangcd(b, a % b, x1,
y1);
    x = y1;
    y = x1 - y1 * (a / b);
    return d;
}

//priority queue comperator works ulta from
vector comperator
struct pqcmp {
    bool operator()(
        pair<int, int> const& a,
        pair<int, int> const& b)
    const noexcept
    {
        if (a.F < b.F)return true;
        else if (a.F == b.F) {
            if (a.S > b.S) return true;
        }
        return false;
    }
};

#pragma        GCC optimize("O3")
#pragma        GCC target("popcnt")


ll accurateFloor(ll a, ll b) {   //atleast one
value has to be positive;
    ll val = a / b;
    while ((val * b) > a) {
        val--;
        deb(val * b);
    }
    return val;
}

__int128 read() {
      __int128 x = 0, f = 1;
      char ch = getchar();
      while (ch < '0' || ch > '9') {
            if (ch == '-') f = -1;
            ch = getchar();
      }
      while (ch >= '0' && ch <= '9') {
            x = x * 10 + ch - '0';
            ch = getchar();
      }
      return x * f;
}
```

```
void print(__int128 x) {
      if (x < 0) {
            putchar('-');
            x = -x;
      }
      if (x > 9) print(x / 10);
      putchar(x % 10 + '0');
}
bool cmp(__int128 x, __int128 y) { return x > y;
}
```

## Promod :

**Sparse table**

```
int k = log2(n) + 1;
int table[n+1][k+1];
for (int j = 1; j <= k; j++) {
    for (int i = 0; i + (1<<j) <= n; i++) {
        table[i][j] =
max(table[i][j-1],table[i+(1<<(j-1))][j-1]);
    }
}
```

**Next int with same set bit**

```
int get_nxt(int x){
    int rightOne;
    int nextHigherOneBit;
    int rightOnesPattern;
    int next = 0;
    rightOne = x & -(signed)x;
    nextHigherOneBit = x + rightOne;
    rightOnesPattern = x ^ nextHigherOneBit;
    rightOnesPattern =
(rightOnesPattern)/rightOne;
    rightOnesPattern >>= 2;
    next = nextHigherOneBit | rightOnesPattern;
    return next;
}
```

**Phi function**

```
vector<int> phi(M + 1);
void phi_1_to_n() {
    phi[0] = 0;
    phi[1] = 1;
    for (int i = 2; i <= M; i++)
        phi[i] = i;

    for (int i = 2; i <= M; i++) {
        if (phi[i] == i) {
            for (int j = i; j <= M; j += i)
                phi[j] -= phi[j] / i;
        }
    }
}


double pi = 2 * acos(0.0);
```

***Template***

```
//RU_tourist
#include<bits/stdc++.h>
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
#include<ext/pb_ds/detail/standard_policies.hpp>

using namespace std;
using namespace __gnu_pbds;

#define ll long long int
#define lll __int128_t
#define ii pair<int,int>
#define pb push_back
#define endl "\n"
#define all(v) v.begin(),v.end()
#define mxpq priority_queue<int>
#define mnpq priority_queue<int, vector<int>,
greater<int>>

template <typename T>
ostream &operator<<(ostream &os, const vector<T>
&v)
{
    os << '{';
    for (const auto &x : v)
        os << " " << x;
    return os << '}';
}

typedef tree<
int,
null_type,
less<int>,
rb_tree_tag,
tree_order_statistics_node_update>
ordered_set;


#define deb(args...){string _s =
#args;replace(_s.begin(), _s.end(), ',', '
');stringstream _ss(_s);istream_iterator<string>
_it(_ss);err(_it, args);}
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...);
}

void solve(){
    cout << "ok\n";
}
```

```
int main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(NULL); cout.tie(NULL);

    #ifndef ONLINE_JUDGE
        freopen("input.txt", "r", stdin);
        freopen("output.txt", "w", stdout);
    #endif

    int test = 1;
    cin >> test;
    for (int tc = 1; tc <= test; tc++) {
        // cout << "Case " << tc << ": ";
        solve();
    }
    return 0;
}
```