1. Perfect Number - <https://codeforces.com/problemset/problem/919/B>

#include<bits/stdc++.h>

using namespace std;

int main() {

    int n;

    cin >> n;

    int count = 0;

    for (int i = 19; ; i++) {

        int temp = i, sum = 0;

        while (temp) {

            sum += temp % 10;

            temp /= 10;

        }

        if (sum == 10) count++;

        if (count == n) {

            cout << i << endl;

            break;

        }

    }

}

1. Aggressive cows - <https://vjudge.net/problem/SPOJ-AGGRCOW>

#include <bits/stdc++.h>

using namespace std;

bool canPlaceCows(vector<int> &stalls, int cows, int minDist) {

    int count = 1;  // First cow placed at first stall

    int lastPos = stalls[0];

    for (int i = 1; i < stalls.size(); i++) {

        if (stalls[i] - lastPos >= minDist) {

            count++;

            lastPos = stalls[i];

        }

        if (count >= cows) return true;

    }

    return false;

}

int aggressiveCows(vector<int> &stalls, int cows) {

    sort(stalls.begin(), stalls.end());

    int low = 1;  // minimum possible distance is at least 1

    int high = stalls.back() - stalls[0];

    int ans = 0;

    while (low <= high) {

        int mid = low + (high - low) / 2;

        if (canPlaceCows(stalls, cows, mid)) {

            ans = mid;

            low = mid + 1;

        } else {

            high = mid - 1;

        }

    }

    return ans;

}

int main() {

    int t;

    cin >> t;

    while (t--) {

        int n, c;

        cin >> n >> c;

        vector<int> stalls(n);

        for (int i = 0; i < n; i++) {

            cin >> stalls[i];

        }

        int result = aggressiveCows(stalls, c);

        cout << result << endl;

    }

    return 0;

}

1. Chat Ban - <https://vjudge.net/problem/CodeForces-1612C>

Newton’s School =>

#include<bits/stdc++.h>

using namespace std;

#define ll long long

int main()

{

    ios\_base::sync\_with\_stdio(false);

    cin.tie(NULL);

    cout.tie(NULL);

    ll t, k, x, ans, l, c;

    long double x1, ans1;

    cin>>t;

    for(;t--;)

    {

        cin>>k>>x;

        c=k\*k;

        if(c<=x){

            cout<<2\*k-1<<"\n";

            continue;

        }

        c=(k\*(k+1))/2;

        if(c<=x){

            x=x-c;

            x=(k\*(k-1))/2-x;

            x1=x;

            ans1=floor((sqrt(1+8\*x1)-1)/2);

            ans=2\*k-1-ans1;

            cout<<ans<<"\n";

        }else{

            x1=x;

            ans1=ceil((sqrt(1+8\*x1)-1)/2);

            ans=ans1;

            cout<<ans<<"\n";

        }

    }

}

Another Approach 🡺

#include <iostream>

using namespace std;

typedef long long ll;

ll emotesSent(ll k, ll m) {

    if (m <= k) {

        return m \* (m + 1) / 2;

    } else {

        ll full = k \* (k + 1) / 2;

        ll rem = m - k;

        ll tail = k - 1;

        ll last = tail - rem + 1;

        return full + (tail + last) \* rem / 2;

    }

}

ll solve(ll k, ll x) {

    ll low = 1, high = 2 \* k - 1, ans = 2 \* k - 1;

    while (low <= high) {

        ll mid = (low + high) / 2;

        if (emotesSent(k, mid) >= x) {

            ans = mid;

            high = mid - 1;

        } else {

            low = mid + 1;

        }

    }

    return ans;

}

int main() {

    int t;

    cin >> t;

    while (t--) {

        ll k, x;

        cin >> k >> x;

        cout << solve(k, x) << "\n";

    }

    return 0;

}

**⏱️ Time Complexity**

**Per Test Case:**

* All operations are constant-time except for sqrt() and some arithmetic.
* So: **O(1)** per test case

1. **K-th Not Divisible by n -** [**https://vjudge.net/problem/CodeForces-1352C**](https://vjudge.net/problem/CodeForces-1352C)

Approach :

1 2 ‘3’ 4 5 ‘6’ 7 8 ‘9’ 10 11 12

n=3

k=7

🡪 10 = k + x

So, we have to find, x = ?

We can see the pattern/logic , like this => ( such as, ( 7 + 3 )/3 = 3.3333… ~ 3 )

x = (k + x) / n

=> nx = k + x;

=> x = k / (n - 1)

For,

1st test case : n=3, k=7

( 7+3 ) % 3 != 0

ans = 7 + 7 / (3-1)

2nd test case : n=4, k=12

( 12+4 ) % 4 == 0

ans = 12 + 12 / (4-1) – 1

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

int main(){

    ll t;cin>>t;

    while (t--)

    {

        ll n, k;cin>>n>>k;

        ll x = k/(n-1);

        if((k+x)%n == 0){

            cout<<k+x-1<<endl;

        }else{

            cout<<k+x<<endl;

        }

    }

    return 0;

}

**Time Complexity**

* For each test case:
  + Division, modulo, addition, and conditional check: all are **O(1)** operations.

Let:

* t = number of test cases

Then:

* **Total Time Complexity:** O(t)

**Space Complexity**

* No extra data structures used.
* Just a few variables per test case.

So:

* **Total Space Complexity:** O(1)

**E- 10474 Where is the Marble? –** [**https://vjudge.net/problem/UVA-10474**](https://vjudge.net/problem/UVA-10474)

[**https://onlinejudge.org/external/104/10474.pdf**](https://onlinejudge.org/external/104/10474.pdf)

#include <bits/stdc++.h>

using namespace std;

int main() {

    int N, Q;

    int caseNum = 1;

    while (cin >> N >> Q, N || Q) {

        vector<int> marbles(N);

        for (int i = 0; i < N; ++i) {

            cin >> marbles[i];

        }

        // Sort the marbles

        sort(marbles.begin(), marbles.end());

        cout << "CASE# " << caseNum++ << ":\n";

        while (Q--) {

            int query;

            cin >> query;

            // lower\_bound returns an iterator to the first element >= query

            auto it = lower\_bound(marbles.begin(), marbles.end(), query);

            if (it != marbles.end() && \*it == query) {

                // +1 for 1-based index

                cout << query << " found at " << (it - marbles.begin() + 1) << '\n';

            } else {

                cout << query << " not found\n";

            }

        }

    }

    return 0;

}

**⏱ Time Complexity**

* For each test case:
  + Sorting marbles: O(N log N)
  + Query processing: Q × O(log N) due to binary search

Overall:

* Efficient even for maximum constraints (N, Q ≤ 10,000).