

Lab Report - 02

Problem : A - Morning

Thought Process : The key observation is that the lock dial follows a **linear pattern: $1 \rightarrow 2 \rightarrow \dots \rightarrow 9 \rightarrow 0$** . Smilo starts at '1', and each digit is dialed by moving forward or backward along this pattern. '0' is treated as 10 to maintain this linear order. For the first digit, the time taken is the absolute distance from '1' to that digit. For each subsequent digit, the movement cost is the absolute difference between current and previous digits (after converting '0' to 10), plus 1 unit for pressing the key. The total sum represents the **minimum time** to enter the 4-digit code.

Source Code :

```
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
void fast() {
    ios::sync_with_stdio(false);
    cin.tie(0);
}

void solve(ll tc) {
    string s;
    cin >> s;
    ll ans = 0;
    for (int i = 0; i < 4; i++) {
        if (i == 0){
            if (s[i] == '0') ans += s[i] - '0' + 10;
            else ans += s[i] - '0';
        }
        else {
            int a = s[i] - '0';
            int b = s[i - 1] - '0';
            if (a == 0) a = 10;
            if (b == 0) b = 10;
            ans += abs(a - b);
            ans++;
        }
    }
    cout << ans << '\n';
}

int main() {
    fast();
    ll t;
    cin >> t;
    while (t--) solve(t);
}
```

Input :


```
10
1111
1236
1010
```

1920
9273
0000
7492
8543
0294
8361

Output :

4
9
31
27
28
13
25
16
33
24

Verdict :

★  priashisg (Priashis)'s submissions for problem A



RunID	Status	Memory	Time	Language	Length	Submit Time
61612126	Accepted	0	31	GNU G++17 7.3.0	2693	0:30:21

Problem : B - Ordinary Numbers

Thought Process : The **key observation** is that all valid numbers consist of **repeated digits**, like 1, 11, 111, 2, 22, 222, and so on — numbers where all digits are the same. Instead of checking each number up to n , we can **generate** these special numbers directly. For each digit i from 1 to 9, we repeatedly append i to itself (e.g., i , $i*10 + i$, $i*100 + i*10 + i$) and stop when the number exceeds n . Each such valid number is counted. This avoids unnecessary computation and ensures we only process numbers that meet the pattern, resulting in an efficient solution.

Source Code :

```
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
void fast() {
    ios::sync_with_stdio(false);
    cin.tie(0);
}
void solve() {
    ll n;
    cin >> n;
    ll ans = 0;
    for (ll i = 1; i <= 9; i++) {
        ll num = i;
        while (num <= n) {
            ans++;
        }
    }
}
```

```

        num = num * 10 + i;
    }
}
cout << ans << '\n';
}
int main() {
    fast();
    ll t;
    cin >> t;
    while (t--) solve();
}

```

Input :

```

6
1
2
3
4
5
100

```

Output :

```

1
2
3
4
5
18

```

Verdict :

★🤖 priashisg (Priashis)'s submissions for problem B

×

RunID	Status	Memory	Time	Language	Length	Submit Time
61614835	Accepted	0	77	GNU G++17 7.3.0	2338	1:28:01

Problem : C - GCD Problem

Thought Process:

Since $\gcd(a, b) = c$, write $a = c \times x$ and $b = c \times y$ with $\gcd(x, y) = 1$. Then $a + b + c = c(x + y + 1) = n$. Choose c so that n is divisible by c , then pick co-prime x, y satisfying $x + y + 1 = \frac{n}{c}$. Selecting appropriate x and y ensures a, b, c are distinct positive integers meeting the conditions.

Source Code :

```

#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
void fast() {
    ios::sync_with_stdio(false);
    cin.tie(0);
}
void solve() {

```

```

int n;
cin >> n;
if (n % 2 == 0) cout << n - 3 << " 2 1\n";
else {
    int x = (n - 1) / 2;
    if (x % 2 == 0) cout << x - 1 << ' ' << x + 1 << " 1\n";
    else cout << x - 2 << ' ' << x + 2 << " 1\n";
}
}
int main() {
    fast();
    int t;
    cin >> t;
    while (t--) solve();
}

```

Input :

```

6
18
63
73
91
438
122690412

```


Output :

```

6 9 3
21 39 3
29 43 1
49 35 7
146 219 73
28622 122661788 2

```

Verdict :

★  priashisg (Priashis)'s submissions for problem C

×

RunID	Status	Memory	Time	Language	Length	Submit Time
61615708	Accepted	100	93	GNU G++17 7.3.0	2372	1:51:01

Problem : D - Fireworks

Thought Process:

The key observation is that both fireworks will be in the air together at time $T = \text{LCM}(a, b)$, and they'll disappear at $T + m + 1$. During the interval $[T, T + m]$, new fireworks appear every a and b seconds. So, the total number of fireworks seen during this time is:

$$\left\lfloor \frac{m}{a} \right\rfloor + \left\lfloor \frac{m}{b} \right\rfloor + 2$$

The `+2` accounts for the two that were already in the air at time T .

Source Code :

```

#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
void fast(){
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
}
void solve(){
    ll a, b, m;
    cin >> a >> b >> m;
    ll ans = 2 + (m / a) + (m / b);
    cout << ans << '\n';
}
int main(){
    int t;
    cin >> t;
    while (t--) solve();
}

```

Input :

```

6
6 7 4
3 4 10
7 8 56
5 6 78123459896
1 1 1
1 1 1000000000000000000

```

Output :

```

2
7
17
28645268630
4
2000000000000000002

```

Problem : F - Long Multiplication

Thought Process: The key observation is that to maximize the product of two numbers formed from the digits of a given number, **we should minimize the difference between them**. To achieve this, we sort the digits and distribute them in such a way that the larger digits are placed alternately—starting with the most significant digit going to one number, and the next largest to the other—ensuring both numbers remain as close as possible in value. This approach helps in achieving a near-balanced product.

Source Code :

```

#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
void fast(){

```

```

ios::sync_with_stdio(false);
cin.tie(nullptr);
}
void solve() {
    string a, b;
    cin >> a >> b;
    ll n = a.size();
    bool swapped = false;
    for (ll i = 0; i < n; ++i) {
        if (!swapped && a[i] < b[i]) {
            swap(a[i], b[i]);
            swapped = true;
        }
        else if (!swapped && a[i] > b[i]) swapped = true;
        else if (swapped && a[i] > b[i]) swap(a[i], b[i]);
    }
    cout << a << '\n' << b << '\n';
}
int main(void){
    fast();
    int t;
    cin >> t;
    while (t--) solve();
}

```

Input :

```

3
73
31
2
5
3516
3982

```

Output:

```

71
33
5
2
3912
3586

```

Verdict :

★ priashisg (Priashis)'s submissions for problem F						
RunID	Status	Memory	Time	Language	Length	Submit Time
61865550	Accepted	0	46	GNU G++17 7.3.0	2530	8:10:29:16

Problem : I - Sum

Thought Process: Check all three possibilities—whether $a = b + c$, or $b = a + c$, or $c = a + b$. If any one is true, print "YES" ; otherwise, print "NO" . This directly verifies if one number equals the sum of the other two.

Source Code :

```

#include <bits/stdc++.h>
using namespace std;

void fast() {
    ios::sync_with_stdio(false);
    cin.tie(0);
}

void solve() {
    int a, b, c;
    cin >> a >> b >> c;
    if (a + b == c || a + c == b || b + c == a) cout << "YES\n";
    else cout << "NO\n";
}

int main() {
    int t;
    cin >> t;
    while (t--) solve();
}

```

Input :

```

7
1 4 3
2 5 8
9 11 20
0 0 0
20 20 20
4 12 3
15 7 8

```

Output :

```

YES
NO
YES
YES
NO
NO
YES

```

Verdict :

★  priashisg (Priashis)'s submissions for problem I

×

RunID	Status	Memory	Time	Language	Length	Submit Time
61614507	Accepted	0	46	GNU G++17 7.3.0	2324	1:19:30