Sum Of Digits

School Accuracy: 67.08% Submissions: 32K+ Points: 0

Given a number, **N**. Find the sum of all the digits of N

Example 1:

Input:
N = 12
Output:
3
Explanation:
Sum of 12's digits:
1 + 2 = 3

Example 2:

Input:
N = 23
Output
5
Explanation:
Sum of 23's digits:
2 + 3 = 5

Your Task:
You don't need to read input or print anything. Your task is to complete the function **sumOfDigits()** which takes an integer **N** as input parameters and returns an integer, total sum of digits of N.

Expected Time Complexity: $O(\log_{10}N)$
Expected Space Complexity: $O(1)$

Constraints:
$1<=N<=10^5$

```cpp
/*
Q1 - Sum of Digits
Problem link: https://www.geeksforgeeks.org/problems/sum-of-digits1742/1
*/

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

//Solution Code
class Solution{
public:
    int sumOfDigits(int N){
        int ans = 0;

        while(N>0) {
            /*
            At each step extract the last digit of N (N%10),
            and it to ans,
            and remove the extracted digit from N (N/10).
            */
            ans += N%10;
            N /= 10;
        }

        return ans;
    }
};

//Driver code (Ignore)
int main() {
    int n; cin>>n;
    Solution ob;
    cout<<ob.sumOfDigits(n)<<endl;
    return 0;
}
```

258. Add Digits

Easy | Topics | Companies | Hint

Given an integer `num`, repeatedly add all its digits until the result has only one digit, and return it.

Example 1:

```
Input: num = 38
Output: 2
Explanation: The process is
38 --> 3 + 8 --> 11
11 --> 1 + 1 --> 2
Since 2 has only one digit, return it.
```

Example 2:

```
Input: num = 0
Output: 0
```

Constraints:

- `0 <= num <= 2^31 - 1`

Follow up: Could you do it without any loop/recursion in `O(1)` runtime?

```cpp
/*
Q2 - Add Digits
Problem Link: https://leetcode.com/problems/add-digits/
*/

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

//Solution Code
class Solution {
public:
    int addDigits(int num) {
        while(num>=10) {
            /*
            Find current digit sum at each step until
            digit sum reaches a value <= 9.
            */
            int currSum = 0;
            while(num>0) {
                currSum += num%10;
                num /= 10;
            }
            num = currSum;
        }
        return num;
    }
};

//Driver Code (Ignore)
int main() {
    int n; cin>>n;
    Solution ob;
    cout<<ob.addDigits(n)<<endl;
    return 0;
}
```

A. Plus or Minus

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given three integers a, b, and c such that **exactly one** of these two equations is true:

- $a + b = c$
- $a - b = c$

Output + if the first equation is true, and − otherwise.

Input

The first line contains a single integer t $(1 \le t \le 162)$ — the number of test cases.

The description of each test case consists of three integers a, b, c $(1 \le a, b \le 9$, $-8 \le c \le 18)$. The additional constraint on the input: it will be generated so that **exactly** one of the two equations will be true.

Output

For each test case, output either + or − on a new line, representing the correct equation.

Example

input

```
11
1 2 3
3 2 1
2 9 -7
3 4 7
1 1 2
1 1 0
3 3 6
9 9 18
9 9 0
1 9 -8
1 9 10
```

output

```
+
-
-
+
+
-
+
+
-
-
+
```

Note

In the first test case, $1 + 2 = 3$.

In the second test case, $3 - 2 = 1$.

In the third test case, $2 - 9 = -7$. Note that c can be negative.

```cpp
/*
Q3 - Plus or Minus
Problem Link: https://codeforces.com/problemset/problem/1807/A
*/

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

int main() {
    int t; cin>>t;
    while(t--) {
        int a,b,c; cin>>a>>b>>c;
        if(a+b == c) {
            cout<<'+'<<endl;
        } else {
            cout<<'-'<<endl;
        }
    }
    return 0;
}
```

263. Ugly Number

Easy | Topics | Companies

An **ugly number** is a positive integer whose prime factors are limited to `2`, `3`, and `5`.

Given an integer `n`, return `true` *if* `n` *is an* ***ugly number***.

Example 1:

```
Input: n = 6
Output: true
Explanation: 6 = 2 × 3
```

Example 2:

```
Input: n = 1
Output: true
Explanation: 1 has no prime factors, therefore all of its prime factors are
limited to 2, 3, and 5.
```

Example 3:

```
Input: n = 14
Output: false
Explanation: 14 is not ugly since it includes the prime factor 7.
```

Constraints:

- $-2^{31} <= n <= 2^{31} - 1$

```cpp
/*
Q4 - Ugly Number
Problem Link: https://leetcode.com/problems/ugly-number/
*/

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

//Solution Code
class Solution {
public:
    bool isUgly(int n) {
        /*
        Return false if n <= 0 beause question specifies
        ugly number is a positive integer.
        */
        if(n<=0) return false;
        while(n%2==0) {
            n /= 2;
        }
        while(n%3==0) {
            n /= 3;
        }
        while(n%5==0) {
            n /= 5;
        }

        return (n==1);
    }
};

//Driver Code (Ignore)
int main() {
    int n; cin>>n;
    Solution ob;
    if(ob.isUgly(n)) cout<<"true"<<endl;
    else cout<<"false"<<endl;
    return 0;
}
```

A. Sum of Round Numbers

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

A positive (strictly greater than zero) integer is called *round* if it is of the form `d00...0`. In other words, a positive integer is round if all its digits except the leftmost (most significant) are equal to zero. In particular, all numbers from 1 to 9 (inclusive) are round.

For example, the following numbers are round: 4000, 1, 9, 800, 90. The following numbers are **not** round: 110, 707, 222, 1001.

You are given a positive integer n ($1 \le n \le 10^4$). Represent the number n as a sum of round numbers using the minimum number of summands (addends). In other words, you need to represent the given number n as a sum of the least number of terms, each of which is a round number.

Input

The first line contains an integer t ($1 \le t \le 10^4$) — the number of test cases in the input. Then t test cases follow.

Each test case is a line containing an integer n ($1 \le n \le 10^4$).

Output

Print t answers to the test cases. Each answer must begin with an integer k — the minimum number of summands. Next, k terms must follow, each of which is a round number, and their sum is n. The terms can be printed in any order. If there are several answers, print any of them.

Example

input Copy

```
5
5009
7
9876
10000
10
```

output Copy

```
2
5000 9
1
7
4
800 70 6 9000
1
10000
1
10
```

```cpp
/*
Q5 - Sum of Round Numbers
Problem Link: https://codeforces.com/contest/1352/problem/A
*/

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

int main() {
    int t; cin>>t;
    while(t--) {
        int n; cin>>n;
        int currPowOfTen = 1;
        /*
        vector is an array of variable size in C++.
        It will be covered later in the course.
        */
        vector<int> ans;
        while(n>0) {
            int dig = n%10;
            n /= 10;
            if(dig>0) {
                ans.push_back(dig*currPowOfTen);
            }
            currPowOfTen *= 10;
        }
        cout<<ans.size()<<endl;
        for(int i : ans) {
            cout<<i<<" ";
        }
        cout<<endl;
    }
    return 0;
}
```

2427. Number of Common Factors

Easy | Topics | Companies | Hint

Given two positive integers `a` and `b`, return *the number of* ***common*** *factors of* `a` *and* `b`.

An integer `x` is a **common factor** of `a` and `b` if `x` divides both `a` and `b`.

Example 1:

```
Input: a = 12, b = 6
Output: 4
Explanation: The common factors of 12 and 6 are 1, 2, 3, 6.
```

Example 2:

```
Input: a = 25, b = 30
Output: 2
Explanation: The common factors of 25 and 30 are 1, 5.
```

Constraints:

- `1 <= a, b <= 1000`

```cpp
/*
Q6 - Number of Common Factors
Problem Link: https://leetcode.com/problems/number-of-common-factors/
*/

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

//Solution Code
class Solution {
public:
    int commonFactors(int a, int b) {
        int ans = 0;
        for(int i=1 ; i<=min(a,b) ; i++) {
            /*
            If i can divide both a and b, then i is a common factor
            */
            if(a%i==0 && b%i==0) ans++;
        }
        return ans;
    }
};

//Driver Code (Ignore)
int main() {
    int a,b; cin>>a>>b;
    Solution ob;
    cout<<ob.commonFactors(a,b)<<endl;
    return 0;
}
```