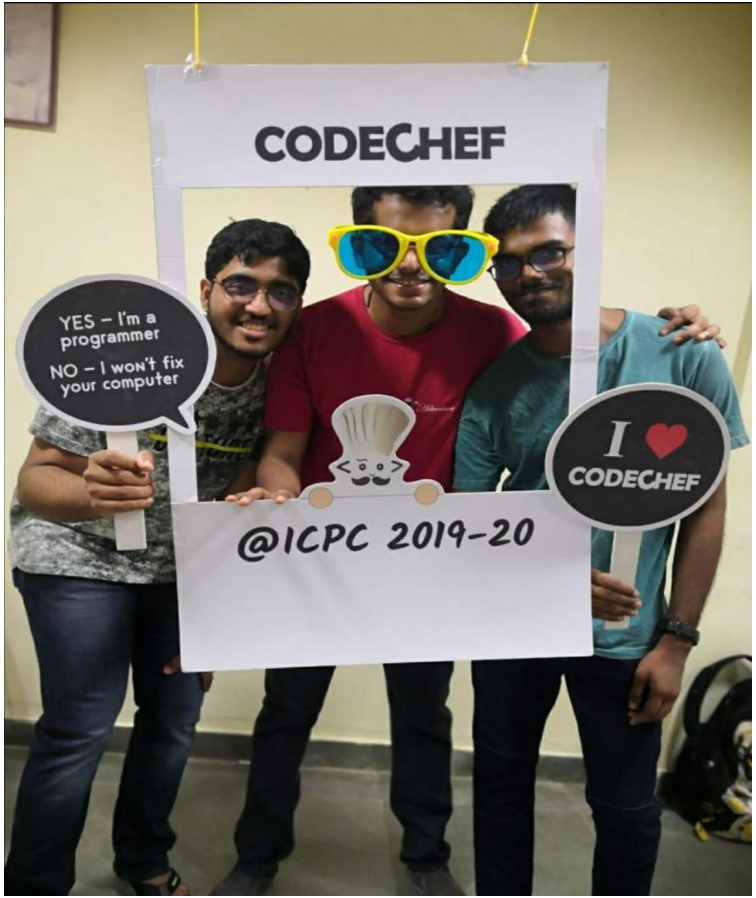
Competitive Coding

By Ashley Lobo

Who Am I?

- I am a Final Year Student @ Fr CRCE
- I am a Competitive Programming enthusiast active on different platforms such as Uva Online Judge, Codechef, Codeforces etc.
- I go by the username : ashlobo247
- I have participated in various Coding Competitions such as ACM ICPC, Google CodeJam etc.



How will be go about in the next 1 hour

- Introduction
- Basic Guidelines
- Problem Analysis
- Ad-Hoc Problems and Complete Search
- Greedy and Dynamic Programming

Introduction

- Using algorithms and data structures to solve problems.
- Google CodeJam, Facebook Hackercup, Google Kickstart, ACM ICPC are few well known competitions.
- Popular sites include Hackerrank, Codechef, Codeforces, AtCoder etc
- Preferred Languages are C++, Java, Python

Introduction

- Know your data structures very well (Complexity, Syntax). C++ has STL which has most of the data structures implementations.
- Practice from sites such as Hackerrank, Codechef Codeforces.
- Attempt for contests both online as well as offline. Short contest preferred.(Codechef, Codeforces, AtCoder have a lot of contests).
- Aim to solve fast with maximum accuracy.
- Do complexity analysis.
- If you are attempting a live contest, check solutions of problems you have solved and reattempt problems which could not be solved.
- Check out other Successful submissions and see how they approached the problem.

Introduction

- First Year: Best time to start problem solving. CSE students can complement with DS and Algo. I myself started in my 2nd sem.
- Second Year: Good time to start. Nearly a year before placements so will definitely help.
- Third Year: If you are trying for technical placements, it is highly recommended to start coding.
- Fourth Year: If you are interested in technical role, it may help in future.

Basic Guidelines

- 32-bit signed integers (int) have upper limits of approx 2 x 10^9

  0<n<10^6. Here we can safely assign n as an int.

- 64-bit signed integers (long long) have upper limits of approx 9 x 10^18

  0<n<10^12. Here we cannot assign n as an int.

  We have to use data type long long.

Basic Guidelines ..contd (Time)

- Approximately 10^8 instructions can be handled by today's computers in few seconds

So if time limit for program is 1 second.
0<n< 10^6

```
for (int i=0 ; i<n ; i++)  // looping from 1 to n
        for (int j=0 ; j<n ; j++)    // nested loop from 1 to n
                printf("hello\n");
```

Hello will be printed n*n times which is max 10^12 instructions

Basic Guidelines ..contd (Time)

- Approximately 10^8 instructions can be handled by today's computers in few seconds

So if time limit for program is 1 second.
0<n< 10^6

```
for (int i=0 ; i<n ; i++)  // looping from 1 to n
        printf("hello\n");
```

Hello will be printed n times which is max 10^6 instructions

Basic Guidelines ..contd (Memory)

- Approximately 4 MB is space taken by integer array of size of 10^6 Or 2d array of 10^3 *10^3.
- Memory declaration inside a function : 10^6
- Larger memory declaration : approx 10^8 can be done globally

```
const n=10^8;
int arr[n];
int main ( ){
        int arr[10^6];
        int arr[10^8];
}
```

Basic Guidelines ..contd (Memory)

- Approximately 4 MB is space taken by integer array of size of 10^6 Or 2d array of 10^3 *10^3.
- Memory declaration inside a function : 10^6
- Larger memory declaration : approx 10^8 can be done globally

```
const n=10^8;
int arr[n];
int main ( ){
        int arr[10^6];
        int arr[10^8];
}
```

Basic Guidelines ..contd (Complexity)

- A recursive algorithm with x recursive calls per level and has y levels
  The complexity is x^y.
- An iterative algorithm with k nested loops with about n iterations per loop.
  The complexity is n^k

Basic Guidelines ..contd (Errors)

- Common errors
  - WA -Wrong Answer
  - RE- Runtime Error: Invalid mathematical operations, accessing out of scope memory.
  - TLE- Time Limit Exceeded:Run time is more than what is mentioned.
  - CPE- Compile time Error: Error during compilation

Problem Analysis

- Understand the problem
- Look at test cases and see if what you have understood aligns with test case
- Check for constraints
- Based on constraints come up with a rough solution
- If you cannot come up with a solution which is efficient based on constraints given , try writing a brute force and then make it efficient.
- Go through code. Check for any errors or maybe
- Run your code once on test input and see if it matches output

C. K-th Not Divisible by n

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

You are given two positive integers n and k. Print the k-th positive integer that is not divisible by n.

For example, if $n = 3$, and $k = 7$, then all numbers that are not divisible by 3 are: $1, 2, 4, 5, 7, 8, 10, 11, 13 \ldots$. The 7-th number among them is 10.

Input

The first line contains an integer t ($1 \le t \le 1000$) — the number of test cases in the input. Next, t test cases are given, one per line.

Each test case is two positive integers n ($2 \le n \le 10^9$) and k ($1 \le k \le 10^9$).

Output

For each test case print the k-th positive integer that is not divisible by n.

Example

input Copy

```
6
3 7
4 12
2 1000000000
7 97
1000000000 1000000000
2 1
```

output Copy

```
10
15
1999999999
113
1000000001
1
```

Ad Hoc problems

- These are problems where there is no general way to solve these problems.
- It usually involves logical and analytical thinking.
- May include mathematical concepts, string based concepts.
- They may involve basic data structures.
- Improves with practice.

Complete Search problems

- Can be called brute force but with techniques to improve complexity or reduce searching.
- Usually iterative or recursive.
- Some techniques to reduce complexity are
  - Preprocessing data (Prefix array etc)
  - Pre Computing information needed
  - Good use of Data Structures

Greedy

- We choose best available to us.
- Most problems usually involve sorting input.
- At each given step we choose the best choice at given moment

Dynamic Programming

- Dynamic Programming is usually used to solve optimization problems and counting problems.
- In these problems, the cost at each step depends on steps taken previously.
- Recursive Relation -> Memoization -> Tabulation

Dynamic Programming(Memoization)

- Memoization a technique used to create a smart efficient recursive function
- It is used in top down dp.
- We store repeated states to avoid recomputation.

Dynamic Programming(Tabulation)

- Doesn’t contain recursion.
- Used in bottom up dp
- Iteratively fill in states in a dp table.

Resources

- Hackerrank
- Codechef
- Codeforces
- Atcoder(Educational DP)
- Hackerearth
- Uva Uhunt

Thank You :)