[![Stargazers][stars-shield]][stars-url]

[![Forks][forks-shield]][forks-url]

[![LinkedIn][linkedin-shield]][linkedin-url]

[forks-shield]: https://img.shields.io/github/forks/ShahjalalShohag/Competitive-Programming-A-Complete-Guideline.svg?style=for-the-badge

[forks-url]: https://github.com/ShahjalalShohag/Competitive-Programming-A-Complete-Guideline/network/members

[stars-shield]: https://img.shields.io/github/stars/ShahjalalShohag/Competitive-Programming-A-Complete-Guideline.svg?style=for-the-badge

[stars-url]: https://github.com/ShahjalalShohag/Competitive-Programming-A-Complete-Guideline/stargazers

[linkedin-shield]: https://img.shields.io/badge/-LinkedIn-black.svg?style=for-the-badge&logo=linkedin&colorB=555

[linkedin-url]: https://linkedin.com/in/ShahjalalShohag

### Table of Contents

- [Table of Contents](#table-of-contents)

- [Discord Server](#discord-server)

- [What is Competitive Programming(CP)?](#what-is-competitive-programmingcp)

- [Problems and Algorithms](#problems-and-algorithms)

- [What's in a Problem?](#whats-in-a-problem)

- [How to Solve a Problem?](#how-to-solve-a-problem)

- [The Ultimate Question: Why should I start Competitive Programming?](#the-ultimate-question-why-should-i-start-competitive-programming)

- [The Secret](#the-secret)

- [Seriously vs Sincerely](#seriously-vs-sincerely)

- [What Progress is Really Like](#what-progress-is-really-like)

- [Which language should I use for CP?](#which-language-should-i-use-for-cp)

- [Where to write codes? What are the best IDEs?](#where-to-write-codes-what-are-the-best-ides)

- [Sublime Text Setup](#sublime-text-setup)

- [My Windows Build](#my-windows-build)

- [My Mac Build](#my-mac-build)

- [bits/stdc++.h file not found on Mac](#bitsstdch-file-not-found-on-mac)

- [Infinite Loop](#infinite-loop)

- [VS Code Setup](#vs-code-setup)

- [Useful Keyboard Shortcuts](#useful-keyboard-shortcuts)

- [General](#general)

- [Sublime Text Specific](#sublime-text-specific)

- [What is a Contest?](#what-is-a-contest)

- [Competitive Programming Platforms](#competitive-programming-platforms)

- [Verdicts](#verdicts)

- [What to do if I don't know how to solve a problem?](#what-to-do-if-i-dont-know-how-to-solve-a-problem)

- [How to Read Problem Statements?](#how-to-read-problem-statements)

- [How to practice a problem efficiently?](#how-to-practice-a-problem-efficiently)

- [Kidlin's Law](#kidlins-law)

- [Number of Problems](#number-of-problems)

- [Machine Learning](#machine-learning)

- [ICPC and National Contests](#icpc-and-national-contests)

- [FAQ](#faq)

- [Books](#books)

- [Important Notes](#important-notes)

- [Tutorial](#tutorial)

- [Life Hack(if you are from Bangladesh)](#life-hackif-you-are-from-bangladesh)

- [Code Library](#code-library)

- [Common Mistakes](#common-mistakes)

- [How to Debug](#how-to-debug)

- [Stress Testing](#stress-testing)

- [Coding Style](#coding-style)

- [More About Contests](#more-about-contests)

- [Codeforces](#codeforces)

- [AtCoder](#atcoder)

- [CodeChef](#codechef)

- [Should I Participate in Contests?](#should-i-participate-in-contests)

- [How to not Miss any Contest?](#how-to-not-miss-any-contest)

- [Virtual Contests](#virtual-contests)

- [Useful Tools](#useful-tools)

- [\[Contest\]Solve some very basic problems.](#contestsolve-some-very-basic-problems)

- [\[Codes\] Check how to solve the problems of Contest 1](#codes-check-how-to-solve-the-problems-of-contest-1)

- [\[Codes\] Check how to solve the problems of Contest 2](#codes-check-how-to-solve-the-problems-of-contest-2)

- [\[Editorials and Codes\] Check how to solve the problems of Contest 3](#editorials-and-codes-check-how-to-solve-the-problems-of-contest-3)

- [Similar Tutorials](#similar-tutorials)

### Discord Server

Join the best Competitive Programming Discord Server to get any kinda help instantly for free: [BCS - Bangladesh CP Server](https://discord.gg/hDSMZATsrM)

### What is Competitive Programming(CP)?

Writing code to solve problems or tasks is the essence of programming. Competitive programming turns this into a sport, with competitors competing (typically online) to solve a bunch of such problems in a restricted period of time.

"A programming competition generally involves the host presenting a set of logical or mathematical problems, also known as puzzles, to the contestants (who can vary in number from tens to several thousand), and contestants are required to write computer programs capable of solving each problem. Judging is based mostly upon several problems solved and time spent for writing successful solutions." - Wikipedia

### Problems and Algorithms

A problem is often based on arithmetic, logic, and/or \_algorithms\_ and looks somewhat like [this](https://cses.fi/problemset/task/1068). Such challenges often include a statement (detailing the task), the input and output format, and input, time and memory constraints.

<br>

But what is an Algorithm? "An algorithm is a step-by-step procedure or a set of rules for solving a specific problem or accomplishing a specific task. It is a precise and well-defined sequence of instructions that can be followed to solve a problem or perform a computation."

Here's an algorithm for brushing your teeth:

- Gather your toothbrush, toothpaste, and a cup of water.

- Wet your toothbrush with water.

- Squeeze a small amount of toothpaste onto the toothbrush.

- Put the toothbrush in your mouth and scrub each section of your teeth for some time.

- Rinse your mouth with water.

- Rinse your toothbrush with water and put it back in its place.

- Wipe your mouth with a towel if necessary.

Just like a computer algorithm, this is a series of specific steps you follow to accomplish a specific task - in this case, brushing your teeth. If you follow these steps in the right order, you will have successfully cleaned your teeth! This illustrates the core principle of an algorithm: a set of instructions that, when followed in sequence, help you accomplish a task or solve a problem.

<br>

\*\*Problem-Solving is all about applying algorithms to solve problems.\*\*

### What's in a Problem?

- \*\*Problem Statement:\*\* This defines the task that needs to be solved.

- \*\*Constraints:\*\* These indicate the range or size of the input that your program should be able to handle.

- \*\*Input and Output Instructions:\*\* These detail what and how your program should read as input and write as output.

- \*\*Sample Input and Output:\*\* There will be a few sample inputs and outputs to help you understand the problem better.

- \*\*Notes:\*\* These supplement the problem statement by demonstrating scenarios and clarifying the task.

- \*\*Test Cases:\*\* Your program should be able to pass these scenarios to be considered accepted.

- \*\*Limits:\*\* These indicate the time and memory limits that your program should not exceed.

<br>

You are required to submit a solution in a supported programming language of your choice. The solution should execute within the stated time and memory limits.

Your solution will be run against a series of test cases once you submit the solution. The test cases will be \_exactly\_ as the input format stated in the problem statement. To get accepted, your solution must generate the correct output for \*\*all\*\* test cases.

Note that the test cases are hidden and you will not be able to see them before getting accepted. But you can 100% trust that the test cases are correct and follow the input format stated in the problem statement and the constraints are also correct.

<br>

Why do the solutions need to be run against a series of test cases? Because one problem can be solved in many ways and in many languages. So it's impossible to check the correctness of a solution by looking at the code. So the solution is: let's say I have created the problem, I know the correct solution and I have created a series of test cases that will check the correctness of the solution. So when you submit your solution, it will be run against my test cases and if your solution generates the same output as mine, then your solution will be accepted.

And this is why there should be input constraints because a solution may work on integers, but may not work on floating points. For example, if the input is an integer $n$, then the constraint is like this: $1 \le n \le 10^9$. This means that the input will be an integer and it will be between $1$ and $10^9$. So if you are taking the input as an integer, then you should make sure that your program can handle all integers between $1$ and $10^9$.

And this is why there should be time and memory constraints because a solution may work on small inputs, but may not work on large inputs. If your solution takes $1$ year to run, then I won't wait for $1$ year to check the correctness of your solution!

And if your solution takes $100$ GB of memory to run, then I won't buy $100$ GB RAM to check the correctness of your solution! That's why there should be memory constraints.

<br>

### How to Solve a Problem?

Solve this problem: [smash me](https://cses.fi/problemset/task/1068)

1. \*\*Understand the problem statement:\*\* This is crucial. Make sure you understand what the problem is asking for, the constraints you have to work with, and the input and output formats. Misunderstanding any part of the problem can lead to a solution that doesn't work.

2. \*\*Break It Down:\*\* Most problems are easier to solve when broken down into smaller, manageable parts. Identify the different components of the problem and try to understand how they relate to each other.

3. \*\*Make a Plan/Algorithm:\*\* Once you've broken down the problem, you should have a better idea of how to solve it. Formulate a strategy for solving the problem. Use pen and paper to write down the steps you need to take to solve the problem. \*\*You should have a clear idea of your solution before you start writing the code for it.\*\*

4. \*\*Write the Code:\*\* Once you have a solid plan, you can start coding. Write your code carefully, keeping it as clean and clear as possible. This will make it easier to debug and improve later.

5. \*\*Test Your Solution:\*\* Once you've written your code, you should test it against the sample test cases provided in the problem statement. If your solution passes all the sample test cases, you can submit it. You can also try to come up with your own test cases to check your solution.

6. \*\*Debug and Improve:\*\* If your solution doesn't pass all the sample test cases, you should debug it. Debugging means finding the bugs/errors. Go through your code line by line and check if it's doing what you expect it to do. If you find any bugs, fix them and test your solution again. If you're unable to find any bugs, you can ask for help from your friends or the community.

7. \*\*Optimize Your Code:\*\* Once your code is working correctly, see if there's any way to make it more efficient. Can you reduce its time complexity? Can you use less memory? Even if your solution is already acceptable, optimizing your code is good practice and can make a big difference in more challenging problems. We will learn about time and memory complexity later.

8. \*\*Submit Your Solution:\*\* Once you're satisfied with your solution, you can submit it. Even if you're confident in your solution, be prepared for the possibility of it being rejected. If it is, don't be discouraged; use it as an opportunity to learn and improve.

Solve the problem using the above steps. Here is the code for the problem in C++:

```c++

#include<iostream> // header file for taking input output

using namespace std;

int main() {

long long int n; cin >> n; // take the input

while (n != 1) { // loop until n is equal to 1

cout << n << ' ';

if (n % 2 == 0) { // if n is even

n = n / 2; // divide n by 2

}

else { // otherwise, if n is odd

n = 3 \* n + 1; // multiply n by 3 and add 1

}

}

cout << 1 << '\n'; // end the output with printing 1

return 0;

}

// we have used `long long int` instead of `int`, thats because

// inside the loop the number can grow enough big that an `int`

// variable can't store that

```

### The Ultimate Question: Why should I start Competitive Programming?

Well, I believe that the ultimate goal of my existence is to be happy. And I have learned from many well-established people that getting a great job or being famous or other common stereotypical goals won't make you happy as when you get used to those they become meaningless.

I think happiness is living your present with excitement.

That being said, now your main goal is to find something that will make you happy. There are lots of things you can do like being a musician or an artist etc. For me it was CP. I can lose myself in CP for hours and hours and still hold my excitement. A good contest is enough to make my day.

Man, I am not saying that you have to choose CP too. Find anything that suits you well, delve into that and maybe you will find peace.

Because I wanna spend my precious time on something so that in the end I can say with Heisenberg, "I did it for me. I liked it, I was good at it, and I was really... I was alive".

\*\*So what kinda benefits you will get by doing CP?\*\*

- Fun and excitement(tons of it).

- "Competitive programming builds the basics in you. You became so expert in coding anything that learning framework then becomes a very very easy task. Because when you learn to read the codes of hard or advanced algorithms, what can be more complex than those?

CP programming makes you versatile, so you can move from any stack to another. But when you only learn a specific framework or stack, your knowledge gets bounded.

You can compare learning any natural language with this. Let's say you want to learn German/English. You need to learn grammar first then you can write a paragraph. In coding, you can think of CP programming as grammar learning and frameworks as paragraph writing! When you know grammar you can write a paragraph on any topic.

Even after a good or average (not the best) career in CP, you will find the database, distributed system, and machine learning topics very much understandable to you."- Raihat Zaman Neloy

- You will find Interview problems much much easier if you do CP. Problem-solving skill is required in interviews and CP is the best and most exciting way to learn problem-solving.

- "Competitive programming is recognized and supported by several multinational software and Internet companies, such as Google and Meta (Facebook). Several organizations host programming competitions on a regular basis." - Wikipedia

- Participating in contests will help you show off your skills and get you noticed by recruiters. This was the case for me.

- Big Tech companies like Google, Facebook, Amazon, etc. hire people through interviews where in technical interviews (one part of the whole interview process), problem-solving is the main thing they look for and they ask questions that are very similar to CP problems. So if you do CP you will have a great advantage in technical interviews.

- [and many more...](https://www.quora.com/What-have-you-gained-from-competitive-programming-Did-you-go-into-research-Did-it-help-you-in-any-aspect-as-a-software-engineer-Did-it-help-you-get-an-in-depth-knowledge-of-a-programming-language-Did-it-affect-your-problem-solving-skills)

Let's discuss something if you think CP is the thing that you wanna do.

Many of us set this goal like I wanna be red and continuously \*\*looking at the goal\*\* and not enjoying our \*\*current\*\* hard work. I am not red but I can guarantee myself that when I will be red I will be happy for a day or two and will get used to it. So what was my 4-5 years of hard work all about? Just a day of excitement? I don't believe in so. Wouldn't it be great if I could live those 4-5 years of my life with excitement? Well yes. This is what I am currently doing. I am living in the present, working hard and whether I become successful or not I will still be happy as I was \*\*alive\*\* throughout the whole process and lived my life to the fullest.

May you find that something that you have been looking for throughout your life!

### The Secret

Not everyone has the privilege to do the things that they enjoy doing. If you have that privilege, then it's really cool. But most people don't have that privilege. So "Instead of doing things you enjoy, learn to enjoy the things you do". And CP is one of the things that you can easily fall in love with.

### Seriously vs Sincerely

If you find yourself approaching things too seriously like it's no fun playing a game if you take it "too seriously". So if you want to have fun, you should switch to approaching things sincerely, like you are still gonna give it all, but you are gonna recognize that this is a game and you are gonna try and enjoy yourself while doing it. This philosophy also applies to CP. Watch [this](https://www.youtube.com/watch?v=FbSNfj2S6Pw) for more.

### What Progress is Really Like

Imagine that you have an ice cube sitting on the table in front of you.

The room is cold and you can see your breath. It is currently twenty-five degrees. Ever so slowly, the room begins to heat up.

Twenty-six degrees. <br>

Twenty-seven. <br>

Twenty-eight.

The ice cube is still sitting on the table in front of you.

Twenty-nine degrees. <br>

Thirty. <br>

Thirty-one.

Still, nothing has happened.

Then, thirty-two degrees. The ice begins to melt. A one-degree shift, seemingly no different from the temperature increases before it, has unlocked a huge change.

Breakthrough moments are often the result of many previous actions, which build up the potential required to unleash a major change. This pattern shows up everywhere. Cancer spends 80 percent of its life undetectable, then takes over the body in months. Bamboo can barely be seen for the first five years as it builds extensive root systems underground before exploding ninety feet into the air within six weeks.

Credit: The book Atomic Habit by James Clear

So don't get depressed after starting CP. It takes time to get your work reflected on your progress.

### Which language should I use for CP?

Most Competitive Programmers (more than 95%) use C++ mostly because it's quite faster than other languages and it has some great built-in libraries(Standard Template Library (STL)) to ease your life. There are also many [other reasons](https://www.quora.com/Why-do-competitive-programmers-prefer-to-use-C++-instead-of-Java-in-the-programming-contests) for using C++.

You can also use other languages too but for CP C++ is preferable. Languages are just tools and it doesn't take more than a few days to understand the basics of a language.

Note that you don't have to be a master in C++ to start your CP journey. If you know the basics i.e. variables, data types, operators, conditions, functions, and loops, then you are ready to start CP. You can learn them from [here](https://www.javatpoint.com/cpp-tutorial) or any of your favorite youtube channels (for example this [freeCodeCamp C++ Tutorial](https://youtu.be/vLnPwxZdW4Y) is also good).

### Where to write codes? What are the best IDEs?

IDEs are where you will write your codes. You can use Sublime Text, Codeblocks, VS Code, or any other IDE that you like.

I use Sublime Text and it's pretty good \*\*for CP.\*\*

#### Sublime Text Setup

1. \*\*Setup:\*\* Check [this video](https://www.youtube.com/watch?v=Zlx7gmt3lBU) to set up C++ in Sublime Text for Competitive Programming in \*\*Windows, Mac and Linux.\*\*

2. \*\*Snippets:\*\* Check [this](https://github.com/the-hyp0cr1t3/CC/blob/master/Sublime%20Text%20Setup.md#snippets) to set up some snippets for Sublime Text.

3. \*\*Shortcuts:\*\* Check [this](https://kirankoduru.github.io/python/sublime-text-ninja.html) to find useful shortcuts.

4. [Extra] Useful Tools: Check [this](https://github.com/the-hyp0cr1t3/CC/blob/master/Sublime%20Text%20Setup.md) to set up useful tools (although not needed normally, I also don't use these).

##### My Windows Build

My sublime build system for CP looks like this on Windows:

```c++

{

"shell\_cmd": "g++ -Wl,-stack=268435456 -std=c++17 \"${file}\" -o \"${file\_path}/${file\_base\_name}\" && \"${file\_path}/${file\_base\_name}\" < \"C:\\Sublime\_Inout\\input.txt\" > \"C:\\Sublime\_Inout\\output.txt\"",

"file\_regex": "^(..[^:]\*):([0-9]+):?([0-9]+)?:? (.\*)$",

"working\_dir": "${file\_path}",

"selector": "source.c++",

}

```

Here, change `C:\\Sublime\_Inout\\input.txt\` by the exact path of your input file and change `C:\\Sublime\_Inout\\output.txt\` by the exact path of your output file.

Check [this](https://www.youtube.com/watch?v=G4qyzclix10) to know how to get file paths on Windows.

##### My Mac Build

My sublime build system for CP looks like this on Mac:

```c++

{

"shell\_cmd": "g++-11 -Wl,-stack\_size,0x20000000 -std=c++17 \"${file}\" -o \"${file\_path}/${file\_base\_name}\" && \"${file\_path}/${file\_base\_name}\"<~/Desktop/Codes/input.txt>~/Desktop/Codes/output.txt && rm \"${file\_path}/${file\_base\_name}\"",

"file\_regex": "^(..[^:]\*):([0-9]+):?([0-9]+)?:? (.\*)$",

"working\_dir": "${file\_path}",

"selector": "source.c++",

}

```

Here, change `~/Desktop/Codes/input.txt` by the exact path of your input file and change `~/Desktop/Codes/output.txt` by the exact path of your output file.

Check [this](https://www.youtube.com/watch?v=gxU1wlBAsig) to know how to get file paths on Mac.

Make sure to install `g++-11` using `brew install gcc@11`. Run `g++-11 --version` to check if it is installed properly.

<br>

<details> <summary> Your sublime text should look something like this </summary>

![sublime\_text](https://i.ibb.co/3MvXcZ3/image.png)

</details>

<br>

##### bits/stdc++.h file not found on Mac

In Mac, if you get `bits/stdc++.h file not found` error, check [this](https://shahjalalshohag.notion.site/How-to-Fix-bits-stdc-h-file-not-found-in-MacOS-aa9e384cc6014bdc93769e7d110c469d) out. This should fix the issue, if not, then contact me.

##### Infinite Loop

When you are running your code in sublime text, if you get an infinite loop, then you can press `Ctrl + C` or `Cmd + C` to terminate your program. Or you can use `Tools > Cancel Build`.

Sometimes, your infinite loop might print an enormous amount of data to the output file in an instant. So your computer might freeze and if nothing of the above works, then you can just open the task manager and terminate the sublime text process. If that doesn't work, then you can just restart your computer and delete the output file.

#### VS Code Setup

Check [this](https://medium.com/@chinmaykulkarni8/how-to-setup-visual-studio-code-for-c-c-java-python-competitive-programming-angular-22fdc9b1f4c6) to setup C, C++, Java, Python in Visual Studio Code for Competitive Programming.

Run the following code to test your sublime text setup:

```c++

#include <bits/stdc++.h>

using namespace std;

int main() {

int a, b;

cin >> a >> b;

cout << a + b << endl;

return 0;

}

```

### Useful Keyboard Shortcuts

Shortcuts for Windows / Mac are given side by side.

#### General

1. `Ctrl + S` / `Cmd + S`: Save - Saves the current file or document.

2. `Ctrl + Shift + S` / `Cmd + Shift + S`: Save As - Saves the current file with a different name or location.

3. `Ctrl + C` / `Cmd + C`: Copy - Copies the selected text or item.

4. `Ctrl + V` / `Cmd + V`: Paste - Pastes the copied or cut text or item.

5. `Ctrl + X` / `Cmd + X`: Cut - Cuts the selected text or item.

6. `Ctrl + Z` / `Cmd + Z`: Undo - Reverses the last action.

7. `Ctrl + Y` / `Cmd + Y`: Redo - Reverses the last undo action.

8. `Ctrl + F` / `Cmd + F`: Find - Opens a search bar to find text within the current document or page.

9. `Ctrl + A` / `Cmd + A`: Select All - Selects all text or items in the current document or window.

10. `Alt + Tab` / `Cmd + Tab`: Switch Application - Switches between open applications or windows.

11. `Ctrl + O` / `Cmd + O`: Open File - Opens a file for editing.

12. `Ctrl + N` / `Cmd + N`: New File - Creates a new file.

#### Sublime Text Specific

1. `Ctrl + P` / `Cmd + P`: Go to File - Allows you to search for a file in the current project.

2. `Ctrl + D` / `Cmd + D`: Multiple Selections - Selects the next occurrence of the current word or selection and allows you to edit them all at once!

3. `Ctrl + /` / `Cmd + /`: Toggle Comment - Comments out the selected code or text.

4. `Ctrl + F5` / `Cmd + B`: Build/Run - Builds or runs the current file.

5. `Ctrl + Shift + D` / `Cmd + Shift + D`: Duplicate - Duplicates the selected text or line.

6. `Ctrl + K + B` / `Cmd + K + B`: Toggle Sidebar - Shows or hides the sidebar.

7. `Ctrl + Shift + Up Arrow` / `Cmd + Shift + Up Arrow`: Move Line Up - Moves the current line or selection up by one line.

8. `Ctrl + Shift + Down Arrow` / `Cmd + Shift + Down Arrow`: Move Line Down - Moves the current line or selection down by one line.

9. `Ctrl + Shift + N` / `Cmd + Shift + N`: New file - Creates a new file.

Note that you can change the shortcuts in sublime text by going to `Preferences > Key Bindings`.

### What is a Contest?

A contest is a competition where you have to solve a set of problems within a given time limit. You will be given a set of problems and you have to solve as many problems as you can within the given time limit. You will be ranked based on the number of problems you solve and the time you take to solve them. The person who solves the most number of problems in the least amount of time wins the contest.

### Competitive Programming Platforms

CP Platforms are websites where you can participate in online contests.

The most famous CP platform is [Codeforces](https://codeforces.com/). Check [this](https://blog.codingblocks.com/2019/sites-and-tools-for-competitive-programming/) to know about more platforms.

I would suggest participating in online contests on [Codeforces](https://codeforces.com/), [AtCoder](https://atcoder.jp/), and [CodeChef](https://www.codechef.com/) in the beginning.

We will discuss more about contests later.

### Verdicts

Once you submit your code, you will get a verdict based on multiple criteria. The most common verdicts are:

- \*\*Accepted (AC):\*\* Your solution passed all test cases! Good job!

- \*\*Wrong Answer (WA):\*\* Your program gave an incorrect output for a specific test case. As a result, it wasn't executed on the remaining test set. Note that the test cases are hidden, so you won't be able to see the test case on which your program failed.

- \*\*Compilation Error (CE):\*\* Your code failed to compile, likely due to a syntactic error. Solve the error by testing your code locally. Make sure you've selected the correct compiler upon submission.

- \*\*Runtime Error (RE):\*\* A fault occurred during the execution of your program. This could be due to issues like accessing an out-of-bound array index, dividing by zero, and so on.

- \*\*Time Limit Exceeded (TLE):\*\* Your program took more time to run than the specified limit. Note that, the execution time is the maximum time taken by your program to run on any test case. So, if your program is taking too much time on a specific test case, then it will get a TLE verdict.

- \*\*Memory Limit Exceeded (MLE):\*\* Similar to TLE, your program used more memory than the allowed limit.

- \*\*Presentation Error (PE):\*\* Your program ran successfully, and the output is correct, but the \_output format\_ is incorrect. This is usually due to a missing space, newline, or an extra space or newline.

Note that your code first gets compiled and then gets executed. That's why you will get a CE verdict even before it gets executed.

### What to do if I don't know how to solve a problem?

Most problems have editorials/tutorials. The tutorial link is normally attached to the problem statement.

Also, \*\*join this discord server to get help from general people\*\*. Link: [Bangladesh CP Server](https://discord.gg/hDSMZATsrM)

### How to Read Problem Statements?

- Read the problem statement thoroughly. Try to understand what the problem is asking you to do.

- Identify the key information, constraints, and requirements.

- Break the problem down into smaller parts or subproblems.

- Pay attention to input-output limitations and samples provided in the statement.

- Analyze the sample test cases and try to understand the problem better.

- Check out the notes section of the problem statement. It may contain some useful information.

- Use pen and paper to draw stuff to better understand the problem.

- If you are still confused, try to read the problem statement again.

### How to practice a problem efficiently?

- First of all, \*\*deeply\*\* understand what the problem asks you to do.

- Then you should try to solve it by yourself.

- At first glance, it may look like you have no idea what that random alien-made problem is asking you to do. But take your time. Always try to solve the problem using brute force. After that try to make your solution more efficient.

- Ok, so still you have no idea how to solve the problem? Try to look at it from a whole new angle.

- "Keep trying while you have new ideas, then look up the editorial/tutorial after \*\*15+ minutes of being completely stuck\*\*." - Kamil Debowski

- To be more precise, if you think you are getting into the solution, then take more time and try to solve it. But if you have no clue on how to solve it, then what is the point of wasting your valuable time? It will only slow down your improvement process.

- After solving a problem by looking at the editorial/others' codes, \*\*think about why your way of thinking was not correct, and what did you miss\*\*. This is reallyyyy important.

- Time to implement the problem. Try not to use any unnecessary macros. Try to make it more readable. It will help you debug the solution.

- After that read the implementations of some skilled users (searching for some useful tricks or really nice implementations). \*\*This part is really important which will significantly improve your skill.\*\*

- If the problem uses a new idea/trick/algorithm which is a classic one i.e. it might be helpful in the future then try to write that down so that in the future you can easily access it.

- Now, let me ask you a question, what did you learn from this problem?

### Kidlin's Law

Kidlin's law: If you can write the problem down clearly then the matter is already half solved.

This is also the same here in CP. First, understand what the problem asks you to do. Then proceed to solve it. If you understand it correctly, then you are half done. So don't start solving without understanding what the problem demands.

### Number of Problems

"Stop obsessing over the number of hours spent or problems solved. These numbers don't mean shit because the variance is so high and it is very easy to spend a lot of time and solve a lot of problems without learning anything." - Tähvend Uustalu (CF: -is-this-fft).

So the main goal is \*\*to learn new stuff\*\* by solving problems.

### Machine Learning

It's all about feeding your brain more data and making it learn. CP is not like your exams where you will just read some stuff and spit them out in the exams. CP is to think. CP is to populate your brain with new techniques, new ways to solve problems. So it will take time to make your machine(brain) learn and to use it to solve new problems with a high success rate.

It's all about Machine Learning.

### ICPC and National Contests

International Collegiate Programming Contest (ICPC) is like the world cup of Programming. University students from around the world participate in it.

First, some Regional contests happen in every region and the top teams from each region qualify for the World Finals.

National Contests are a similar format contest but it is only for the students of a specific country. Around 5 to 10 National Contests are held in Bangladesh every year.

Format:

- This is a team contest. Each team consists of 3 members.

- One computer for each team.

- 5 hours long contest.

- 9-15 problems.

Learn more [here](https://www.quora.com/What-is-the-ACM-ICPC).

### FAQ

- [What is the key to CP?](https://codeforces.com/blog/entry/47516)

- [Regarding Interviews and Jobs](https://www.quora.com/How-has-competitive-programming-helped-you-get-a-job)

- [CP Or Development (Part 1)](https://www.geeksforgeeks.org/competitive-programming-vs-software-development-where-should-i-invest-my-time/)

- [CP or Development (Part 2)](https://www.quora.com/Which-is-better-to-invest-time-in-competitive-programming-algorithmic-programming-on-websites-like-CodeChef-or-TopCoder-vs-software-development-app-dev-web-dev-etc-Why)

- [CP or Development (Part 3)](https://medium.com/it-paragon/competitive-programming-vs-software-development-where-should-i-invest-my-time-619841d202f3)

### Books

Check [this](https://cses.fi/book/book.pdf) if you want. It contains everything.

### Important Notes

- \*\*Exercise\*\* (at least running) and drink more water. It will surprisingly boost your learning capability.

- \*\*Getting AC is not the final goal, learning something new is the final goal\*\*. Exactly, for this reason, people solve thousands of problems but can't get better. You need to solve harder problems than your current level so that you can learn something new by solving that problem.

- Also, after you solve a problem, try to do it more efficiently if possible, and look at others' solutions. This way you will learn better and become better faster in the long run. Make sure that you truly \_understand\_ and \_feel\_ the solution.

- After solving a problem, \*\*think about why your way of thinking was not optimal\*\*. This is important too.

- \*\*Design before you code.\*\* Think about the solution/algorithm first, then code it. It will save you a lot of time.

- \*\*SOLVE MORE PROBLEMS\*\*.

- Talk to similar-minded people. Make friends who are also interested in CP. It will help you a lot.

- Always ask `Why?`.

- Why is this solution not correct?

- Why is this solution correct?

- Why is this solution optimal?

- Why is this solution not optimal?

- Why did this solution get TLE?

- Why did this work?

- Why do we exist...?

### Tutorial

- [The Ultimate Topic List](https://youkn0wwho.academy/topic-list)

Check the above link. You will find everything categorized there. Then check the [guideline](https://youkn0wwho.academy/topic-list/guideline) page.

So basically start with the basics section. Complete the topics under this section, you will find resources and problems for everything in increasing order of completion.

After you are done with the basics section, select a difficulty and importance (3\\* topics first), then complete them one by one.

Meanwhile, always participate in live contests and solve some random problems from time to time.

Also, you can sometimes try to do rating-wise practice. And make sure that you are not solving too many easy problems. Try to solve problems that are a bit harder than your current level.

\_Do not get overwhelmed by the number of topics. Just start with the basics and keep going. If you are done with the basics, then you are already better than 80% of the people and ready to be an Expert in Codeforces. And, if you are done with the 3\\* topics of easy and medium difficulty, then you can be a Grandmaster in Codeforces once you solve enough problems.\_

---

### Life Hack(if you are from Bangladesh)

If you want a complete guideline like this for EVERYTHING about CP and you are from Bangladesh, then you can check out my academy and enroll in some courses that fits you well.

Link: [YouKn0wWho Academy](https://academy.shahjalalshohag.com/).

---

### Code Library

- [Almost all the important templates that you will need in CP](https://github.com/ShahjalalShohag/code-library)

### Common Mistakes

Check out [[Tutorial] Common Mistakes in Competitive Programming and How to Avoid Them](https://codeforces.com/blog/entry/111217)

### How to Debug

Debugging is the process of finding and fixing bugs (errors) in your code. It's a very important skill that you need to master.

- \*\*Understand Error Messages:\*\* Read the error messages carefully and try to understand what they mean.

For example, if you see a message like this:

```c++

my\_program.cpp: In function ‘int main()’:

my\_program.cpp:4:12: error: ‘x’ was not declared in this scope

```

This tells you that on line 4 of `my\_program.cpp`, you're trying to use a variable x that the compiler doesn't know about.

- \*\*Print and Check:\*\* This is the best way to debug your code. Print the values of variables and check if they are what you expect them to be.

For example, consider calculating the factorial of a number $n$ where $n$ could be up to $20$. Here's a buggy code:

```c++

#include <bits/stdc++.h>

using namespace std;

int main() {

int n;

cin >> n;

int fact = 1;

for (int i = 1; i <= n; i++) {

fac \*= i;

}

cout << fact << '\n';

return 0;

}

```

If you run this, you will get a compilation error because `fac` is not defined. But if you fix that, you will get a wrong answer. So, you need to print the value of `fac` in each iteration to see what's going wrong.

```c++

#include <bits/stdc++.h>

using namespace std;

int main() {

int n;

cin >> n; // input n = 20

int fact = 1;

for (int i = 1; i <= n; i++) {

fact \*= i;

cout << fact << '\n';

}

cout << fact << '\n';

return 0;

}

```

Now, you can see that some of the values of `fact` become negative after $12!$. Why? Because the maximum value that an `int` can store is $2^{31}-1$ which is less than $13!$. So, you need to use a `long long` instead of an `int`.

```c++

#include <bits/stdc++.h>

using namespace std;

int main() {

int n;

cin >> n; // input n = 20

long long fact = 1;

for (int i = 1; i <= n; i++) {

fact \*= i;

}

cout << fact << '\n';

return 0;

}

```

### Stress Testing

- [Find a counter-test](https://ali-ibrahim137.github.io/competitive/programming/2020/08/23/Stress-Testing.html)

### Coding Style

\*\*IMPROVE YOUR CODING STYLE. IT DESCRIBES YOURSELF!\*\*

- It's important because it makes your code more readable and understandable.

- It's important because it makes your code more beautiful.

- It's important because it helps you to debug your code easily.

- It's important because it helps your teammates to understand your code easily.

- It's important because it will help you during interviews.

- It's important because if you don't follow a coding style in your job, you might get fired!

<br>

Tutorial: [link](https://codeforces.com/blog/entry/64218).

\*\*Online formatter:\*\* [link](https://codebeautify.org/cpp-formatter-beautifier), you can use this to check how your ugly codes can be made beautiful.

<br>In short:

- Use proper indentations and spacings.

- Use `const` if you are using the same value multiple times. For example, `const int N = 1e5 + 9` or `const int INF = 1e9 + 9`.

- Use not more than three macros(for example, `#define ll long long` is acceptable).

- After any condition, loop, or function, write the curly bracket on the same line, not on the next line.

- Write variables on the go. `for (int i = 1; i <= n; i++)`. Here `i` has been declared on the go.

- Use meaningful variable/function names.

- Naming Conventions: `snake\_case` or `camelCase` for variables and functions, `PascalCase` for classes and structures. I personally use `snake\_case` for variables and functions.

- `snake\_case`: Words are separated by underscores. For example, `int max\_value\_possible = 0`.

- `camelCase`: Words are separated by capital letters, but the first word starts with a small letter. For example, `int maxValuePossible = 0`.

- `PascalCase`: Words are separated by capital letters, and the first word also starts with a capital letter. For example, `struct MaxValuePossible {}`.

- Comment your code. It helps you to understand your code later.

- Consistency is the key. If you are using `snake\_case` for variables, then use it everywhere. Don't use `snake\_case` for some variables and `camelCase` for others.

<br>You can follow other users on Codeforces and make them friends (click the star button on the right of the username on CF) and check their solutions using the "friends only" button. I like the coding styles of the following users:

- [neal](https://codeforces.com/profile/neal)

- [YouKn0wWho](https://codeforces.com/profile/YouKn0wWho)

- [mango\_lassi](https://codeforces.com/profile/mango\_lassi)

- [jiangly](https://codeforces.com/profile/jiangly)

- [tourist](https://codeforces.com/profile/tourist)

- [Anachor](https://codeforces.com/profile/Anachor)

- [Bruteforceman](https://codeforces.com/profile/Bruteforceman)

- [Others](https://codeforces.com/blog/entry/77865)

### More About Contests

#### Codeforces

Website: [https://codeforces.com/](https://codeforces.com/)

- \*\*How to Use Codeforces\*\*

- Check this guide: [must check](https://codeforces.com/blog/entry/99660).

- \*\*How do Codeforces' contests work?\*\*

- Read this: [must check](https://codeforces.com/blog/entry/456).

- \*\*Problem Rating\*\*

- Each problem has a rating that indicates contestants with which average rating solved this problem previously. Check [this](https://codeforces.com/blog/entry/62865) to know more about this and \*\*how to sort problems based on ratings.\*\*

- \*\*Problem Tags\*\*

- Each problem has some tags that indicate the topics related to the problem. Check [this](https://codeforces.com/blog/entry/14565) to know more about this.

- \*\*Make Sure You Know the Following:\*\*

- How to submit a solution.

- How to see other's solutions.

- How to see the test cases.

- How to see the standings.

- How to see the editorial.

- How to see the discussions.

- How to sort problems based on ratings.

- How to up-solve problems.

- What are different divisions and how to participate in them.

#### AtCoder

Website: [https://atcoder.jp/](https://atcoder.jp/)

The process is similar to Codeforces.

#### CodeChef

Website: [https://www.codechef.com/](https://www.codechef.com/)

The process is similar to Codeforces.

#### Should I Participate in Contests?

Some people think that they will learn some topics first and once they are done with learning, they will participate in contests. But this is a veryyyy bad practice.

\*\*YOU SHOULD PARTICIPATE IN ALL CONTESTS.\*\*

Participating in contests in real-time, trying to solve problems in a limited amount of time, competing with thousands of people all over the world, not being able to solve during the contest, up-solving after the contest, reading editorials, reading other's solutions, reading discussions, etc. will help you a lot in learning new things and improving your skills.

I will \*\*strongly\*\* recommend you participate in the following contests:

- Codeforces Div. 4, Div. 3, and Div. 2 contests.

- AtCoder Beginner Contests.

- CodeChef Starters Contests.

### How to not Miss any Contest?

You can check the contest tabs on the corresponding websites. But the best way to get notified about all the contests:

- Join this server: [BCS - Bangladesh CP Server](https://discord.gg/hDSMZATsrM) and go to `reaction-roles` channel and react to the first message with the corresponding emoji to get notified about all the future contests!

### Virtual Contests

Virtual contests will give you a real-time experience. It enables the users to run past contests in a special mode that would imitate real competition. It feels just like a real contest with real contestants competing alongside the participant who writes a virtual contest. But it won't affect your rating.

For Codeforces, click on `Virtual participation` on the [contest page](https://codeforces.com/contests) to participate in a virtual contest. You can pick your own start time for the virtual contest.

### Useful Tools

- [Chrome/Firefox Extension to analyze Codeforces profiles](https://codeforces.com/blog/entry/93417)

- [Codeforces Practice Tracker — Browser Extension](https://codeforces.com/blog/entry/78203)

- [CF-Predictor](https://chrome.google.com/webstore/detail/cf-predictor/ocfloejijfhhkkdmheodbaanephbnfhn)

- [Codeforces Live Bot (Telegram Bot)](https://codeforces.com/blog/entry/82669)

- [Codeforces Visualizer](https://cfviz.netlify.app/)

### [Contest]Solve some very basic problems.

Get familiar with problem-solving by solving these problems. You won't need to know anything other than the basics of a language to solve them.

\*\*Hint:\*\* [How to Practice?](https://github.com/ShahjalalShohag/Competitive-Programming-A-Complete-Guideline#how-to-practice-a-problem-efficiently) <br>

\*\*Goal:\*\* Solve ALL problems.

- Contest 1: [link](https://codeforces.com/group/MWSDmqGsZm/contest/219158)

- Contest 2: [link](https://codeforces.com/group/MWSDmqGsZm/contest/219432)

- Contest 3: [link](https://vjudge.net/contest/568978)

\*\*Pro Tip:\*\* To check your ranking on the standings page on Vjudge, click on [settings](https://i.ibb.co/x7b1LQ6/image.png) on the contest page and then click on [Show Practice Submissions](https://i.ibb.co/R2CZ1Wg/image.png).

IT IS HIGHLY RECOMMENDED TO READ AND UNDERSTAND THE FOLLOWING CODES EVEN IF YOU SOLVE A PROBLEM ON YOUR OWN.

You will find very detailed commented out solutions here. Try to follow the same coding style (better spacing, better variable naming, modularized and readable codes etc). You may learn lots of new stuff by reading other's solutions.

### [Codes] Check how to solve the problems of Contest 1

<details> <summary> smash me </summary>

<b>1. Problem A</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

char name[100]; // considering the name has at most 100 characters

cin >> name; // take the name as input

cout << "Hello, " << name << '\n'; // output in the correct format

return 0;

}

```

</details>

<br>

<b>2. Problem B</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

// make sure to declare the correct data types for each variable

// use meaningful and readable variable names

// use proper spacing to make your code look cooler (spaces before and after all operators)

int int\_variable;

long long int long\_long\_variable;

char char\_variable;

float float\_variable;

double double\_variable;

cin >> int\_variable >> long\_long\_variable >> char\_variable >> float\_variable >> double\_variable;

cout << int\_variable << '\n';

cout << long\_long\_variable << '\n';

cout << char\_variable << '\n';

cout << float\_variable << '\n';

cout << double\_variable << '\n';

// don't forget to output newlines after each variable

return 0;

}

```

</details>

<br>

<b>3. Problem C</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

// int data type is enough for this problem as values are <= 10^5

// always check the constraints in the problem statement

// you can also use long long here but thats unnecessary and 2 times slower

// because long long is 64 bit but int is 32 bit

// Also use long long only when its necessary, do not overuse it, not a good practice

int x, y; cin >> x >> y;

// output in the EXACT SAME format as the problem suggests

// 5 + 10 = 15 and 5+10=15 are different from each other! Spaces do matter when you output them

cout << x << " + " << y << " = " << x + y << '\n';

// cout << x << "+" << y << "=" << x + y << '\n'; // wrong! because we are not printing enough spaces

// cout<<x<<" + "<<y<<" = "<<x+y<<'\n'; // correct but looks ugly. we are printing enough spaces IN THE OUTPUT but not using spaces IN THE CODE

// typecast to long long as 1 <= x, y <= 10^5, so in the worst case (the case when the product will be the highest)

// x = y = 10^5, so x \* y = 10^5 \* 10^5 = 10^10 and 10^10 is big enough

// that we can't store it using 32 bits (int). So we should use 64 bit data types

// As a rule of thumb, you can store upto 2\*10^9 in int data type

// and upto 9\*10^18 in long long data type

// but the exact value is of course 2^31 - 1 for int and 2^63-1 for long long

cout << x << " \* " << y << " = " << (long long)x \* y << '\n';

// cout << x << " \* " << y << " = " << x \* y << '\n'; // wrong, overflow!

cout << x << " - " << y << " = " << x - y << '\n';

return 0;

}

```

</details>

<br>

<b>4. Problem D</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

// note that the variable names do not have to be the exactly same as the problem statement

// but it is intuitive to use the same variable names

int A, B, C, D; cin >> A >> B >> C >> D; // int is enough as max value is <= 10^5

// in the worst case A = B = 10^5, so A \* B = 10^10

// and 10^10 can't be stored in int,

// - so we typecast it when multiplying A and B

// - we also store it in a variable (AB) that has long long data type

long long AB = (long long)A \* B;

// int AB = (long long) A \* B; // wrong! as AB is in int, so overflow

// long long AB = A \* B; // wrong! as in the right hand side, A and B both are in int

// so A \* B will also be in int, and so it will overflow

// even before storing the result in the AB variable

long long CD = (long long) C \* D;

long long answer = AB - CD;

cout << "Difference = " << answer << '\n';

// cout << "Difference=" << answer << '\n'; // wrong! as not enough space before and after =

// cout << answer << '\n'; // wrong! as not in the same output format as the problem tells us to do

return 0;

}

```

</details>

<br>

<b>5. Problem E</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

#include<iomanip> // for using setprecision

using namespace std;

// its better to use const for variables that do not change

// you can't modify const variables and thats what we want for PI!

const double PI = 3.141592653;

int main() {

// always use double instead of float as double has more bits

// -> means more precisions -> more accurate results

double R; cin >> R;

double area = PI \* R \* R;

// use setprecision to print UPTO 9 digits after the decimal point

// use fixed to print EXACLTLY 9 digits after the decimal point

// so if the result is 1.4569 then using setprecision(9) will output 1.4569

// and using fixed << setprecision(9) will output 1.456900000

// it is always better to use fixed when you use setprecision

cout << fixed << setprecision(9) << area << '\n';

return 0;

}

```

</details>

<br>

<b>6. Problem F</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

// use long long as values are upto 10^18

long long n, m; cin >> n >> m;

int last\_digit\_of\_n = n % 10; // if you divide a number by 10 then the remainder will always be the last digit

int last\_digit\_of\_m = m % 10;

int answer = last\_digit\_of\_n + last\_digit\_of\_m;

cout << answer << '\n';

return 0;

}

```

</details>

<br>

<b>7. Problem G</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int n; cin >> n;

// the formula for the summation of numbers from 1 to n is = n \* (n + 1) / 2

// in the worst case, n = 10^9

// so sum = 10^9 \* (10^9 + 1) / 2 = around 10^18 / 2

// and we can't store it in int, so we need long long here

long long sum = 0;

// we need to typecast in the right hand side as otherwise the right hand side will overflow

sum = (long long) n \* (n + 1) / 2;

cout << sum << '\n'; // always output a newline at the end of the output file

// sum = 0;

// for (int i = 1; i <= n; i++) {

// sum += i;

// }

// cout << sum << '\n';

//

// this is also correct

// but it will give you time limit exceeded verdict

// as in the worst case, n = 10^9 and then the loop will run 10^9 times

// in general it takes around 1s to run 10^8 operations (loops/additions/etc)

// so it will take around 10s to run 10^9 operations

// but the time limit in the problem is 0.25s

// thats why it will give you TLE

// note that you should ALWAYS understand why your code gives TLE

return 0;

}

```

</details>

<br>

<b>8. Problem H</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

#include <cmath> // for using floor, ceil and round functions

using namespace std;

int main() {

int a, b; cin >> a >> b;

int floor\_value = a / b; // c/c++ division operator does floor division by default IF a and b are integers

// int floor\_value = floor((double)a / b); // also corrects

// ceil of a / b = floor of (a + b - 1) / b when a, b >= 1

// for example, ceil of 9 / 5 = floor of (9 + 5 - 1) / 5 = 2

// why is it true? think for yourself

int ceil\_value = (a + b - 1) / b; // great approach as we are not using any double here

// int ceil\_value = ceil((double)a / b); // also corrects, but it uses doubles which we normally should avoid because of precision issues

int round\_value = round((double) a / b);

cout << "floor " << a << " / " << b << " = " << floor\_value << '\n';

cout << "ceil " << a << " / " << b << " = " << ceil\_value << '\n';

cout << "round " << a << " / " << b << " = " << round\_value << '\n';

return 0;

}

```

</details>

<br>

<b>9. Problem I</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int a, b; cin >> a >> b;

if (a >= b) { // notice where I am using spaces to make the code look better!

cout << "Yes\n";

// cout << "YES\n"; // wrong! as everything is case sensitive in general

}

else {

cout << "No\n";

}

return 0;

}

```

</details>

<br>

<b>10. Problem J</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int a, b; cin >> a >> b;

// a is divisible by b means the remainder after dividing a by b is 0

if (a % b == 0 or b % a == 0) { // you can use "or", "||" for the OR operator

// we aren't using the AND operator here as we want the whole condition to be true

// if any of the condition is true

cout << "Multiples\n";

}

else {

cout << "No Multiples\n";

}

return 0;

}

```

</details>

<br>

<b>11. Problem K</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int a, b, c; cin >> a >> b >> c;

int minimum = min(a, min(b, c)); // take the min of b and c first and then minimize it with a

// int minimum = min(min(a, b), c); // also correct

int maximum = max(a, max(b, c));

// print the minimum first as suggested by the problem

cout << minimum << ' ' << maximum << '\n';

return 0;

}

```

</details>

<br>

<b>12. Problem L</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

#include <cstring> // for strings

using namespace std;

int main() {

// assume each name has at most 1000 characters

char first\_name\_first\_person[1000], second\_name\_first\_person[1000];

char first\_name\_second\_person[1000], second\_name\_second\_person[1000];

cin >> first\_name\_first\_person >> second\_name\_first\_person;

cin >> first\_name\_second\_person >> second\_name\_second\_person;

// check if the second names are equal

bool is\_equal = true; // stores if the strings are equal or not

int len\_1 = strlen(second\_name\_first\_person);

int len\_2 = strlen(second\_name\_second\_person);

if (len\_1 != len\_2) { // if the lengths are not equal then the strings can't be equal

is\_equal = false;

}

else {

// now check if all characters are equal in both strings

for (int i = 0; i < len\_1; i++) {

if (second\_name\_first\_person[i] != second\_name\_second\_person[i]) {

is\_equal = false;

break;

}

}

}

if (is\_equal) {

cout << "ARE Brothers\n";

}

else {

cout << "NOT\n";

}

return 0;

}

```

</details>

<br>

<b>13. Problem M</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

char x; cin >> x;

int ascii = (int)x; // convert to ascii value

if (isalpha(x)) {

// same as if ((x >= 'A' and x <= 'Z') or (x >= 'a' and x <= 'z'))

// same as if ((ascii >= 65 and ascii <= 90) or (ascii >= 97 and ascii <= 122))

cout << "ALPHA\n";

if (isupper(x)) {

// same as if ((x >= 'A' and x <= 'Z'))

// same as if ((ascii >= 65 and ascii <= 90))

cout << "IS CAPITAL\n";

}

else {

// similar function: islower(x) -> to check if x is a small letter

cout << "IS SMALL\n";

}

}

else {

// similar function: isdigit(x) -> to check if x is a digit

cout << "IS DIGIT\n";

}

return 0;

}

```

</details>

<br>

<b>14. Problem N</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

char x; cin >> x;

if (x >= 'A' and x <= 'Z') {

// notice that all ascii values of 'A' to 'Z' are one after another from 65 to 90

int ascii\_of\_capital\_X = (int)x;

int ascii\_of\_capital\_A = (int)'A'; // = 65

int position\_of\_capital\_X = ascii\_of\_capital\_X - ascii\_of\_capital\_A;

int ascii\_of\_small\_a = (int)'a'; // = 97

int ascii\_of\_small\_x = ascii\_of\_small\_a + position\_of\_capital\_X;

char small\_x = (char)ascii\_of\_small\_x;

cout << small\_x << '\n';

}

else {

int ascii\_of\_small\_x = (int)x;

int ascii\_of\_small\_a = (int)'a'; // = 97

int position\_of\_small\_x = ascii\_of\_small\_x - ascii\_of\_small\_a;

int ascii\_of\_capital\_A = (int)'A'; // = 65

int ascii\_of\_capital\_X = ascii\_of\_capital\_A + position\_of\_small\_x;

char capital\_X = (char)ascii\_of\_capital\_X;

cout << capital\_X << '\n';

}

// also correct! and short!

// if (x >= 'A' and x <= 'Z') {

// cout << (char)((x - 'A') + 'a') << '\n';

// }

// else {

// cout << (char)((x - 'a') + 'A') << '\n';

// }

// also correct! and perfect!

// if (isupper(x)) {

// cout << (char)tolower(x) << '\n';

// }

// else {

// cout << (char)toupper(x) << '\n';

// }

return 0;

}

```

</details>

<br>

<b>15. Problem O</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int a, b; char s;

// for 7+54

// cin >> a will add all the digits to a until it finds a non digit character

// then it will find the non digit char +, and then cin >> s will trigger itself

// then as s is a char data type, s will be equal to + and then cin >> b will get triggered

// and then the rest of the digits will get added to b

cin >> a >> s >> b;

if (s == '+') {

cout << a + b << '\n';

}

else if (s == '-') {

cout << a - b << '\n';

}

else if (s == '\*') {

cout << a \* b << '\n';

}

else {

cout << a / b << '\n'; // integer division

}

return 0;

}

```

</details>

<br>

<b>16. Problem P</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int x; cin >> x;

// notice that the number is from 1000 to 9999

// so the first digit can be found by dividing the number by 1000

int first\_digit = x / 1000; // floor division

if (first\_digit % 2 == 0) {

cout << "EVEN\n";

}

else {

cout << "ODD\n";

}

// the following is another of way of getting the first digit

// notice that we can take the input in any format we like

// the judging system only cares about what we print in the output file!

//

// char x[100];

// cin >> x;

// int first\_digit = x[0] - '0';

return 0;

}

```

</details>

<br>

<b>17. Problem Q</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

double x, y;

cin >> x >> y;

if (x == 0 and y == 0) cout << "Origem\n";

else if (y == 0 and (x > 0 or x < 0)) cout << "Eixo X\n";

else if (x == 0 and (y > 0 or y < 0)) cout << "Eixo Y\n";

else if (x > 0 and y > 0) cout << "Q1\n";

else if (x < 0 and y > 0) cout << "Q2\n";

else if (x < 0 and y < 0) cout << "Q3\n";

else cout << "Q4\n";

return 0;

}

```

</details>

<br>

<b>18. Problem R</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int age\_in\_days; cin >> age\_in\_days;

int years = age\_in\_days / 365;

cout << years << " years\n";

age\_in\_days = age\_in\_days % 365;

int months = age\_in\_days / 30;

cout << months << " months\n";

age\_in\_days = age\_in\_days % 30;

cout << age\_in\_days << " days\n";

return 0;

}

```

</details>

<br>

<b>19. Problem S</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

double x; cin >> x;

if (0 <= x and x <= 25) cout << "Interval [0,25]\n";

else if (25 < x and x <= 50) cout << "Interval (25,50]\n";

else if (50 < x and x <= 75) cout << "Interval (50,75]\n";

else if (75 < x and x <= 100) cout << "Interval (75,100]\n";

else cout << "Out of Intervals\n";

return 0;

}

```

</details>

<br>

<b>20. Problem T</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int a, b, c; cin >> a >> b >> c;

int minimum = min(a, min(b, c));

int maximum = max(a, max(b, c));

int mid = a + b + c - minimum - maximum; // the other number that is neither min nor max

cout << minimum << '\n' << mid << '\n' << maximum << '\n';

cout << '\n'; // blank line is important as the problem said so

cout << a << '\n' << b << '\n' << c << '\n';

return 0;

}

```

</details>

<br>

<b>21. Problem U</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

float number; cin >> number;

int integer\_part = (int)number; // convert number to integer to get the integer part

if (number == integer\_part) {

cout << "int " << integer\_part << '\n';

}

else {

float decimal\_part = number - integer\_part;

cout << "float " << integer\_part << " " << decimal\_part << '\n';

}

return 0;

}

```

</details>

<br>

<b>22. Problem V</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int a, b; char s;

cin >> a >> s >> b;

if(s == '<') {

if (a < b) cout << "Right\n";

else cout << "Wrong\n";

}

else if (s == '=') {

if (a == b) cout << "Right\n";

else cout << "Wrong\n";

}

else if (s == '>') {

if (a > b) cout << "Right\n";

else cout << "Wrong\n";

}

return 0;

}

```

</details>

<br>

<b>23. Problem W</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int num1, num2, num3;

char sign1, sign2;

cin >> num1 >> sign1 >> num2 >> sign2 >> num3;

if (sign1 == '\*') {

if (num3 == num1 \* num2) cout << "Yes\n";

else cout << num1 \* num2 << '\n';

}

else if (sign1 == '+') {

if (num3 == num1 + num2) cout << "Yes\n";

else cout << num1 + num2 << '\n';

}

else if (sign1 == '-') {

if (num3 == num1 - num2) cout << "Yes\n";

else cout << num1 - num2 << '\n';

}

return 0;

}

```

</details>

<br>

<b>24. Problem X</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int l1, r1, l2, r2; cin >> l1 >> r1 >> l2 >> r2;

int left\_boundary = max(l1, l2);

int right\_boundary = min(r1, r2);

if (left\_boundary <= right\_boundary) {

cout << left\_boundary << ' ' << right\_boundary << '\n';

}

else {

cout << -1 << '\n';

}

return 0;

}

```

</details>

<br>

<b>25. Problem Y</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

// solution explanation: https://youtu.be/6MSIMpWNRRg

int a, b, c, d;

cin >> a >> b >> c >> d ;

a %= 100;

b %= 100;

c %= 100;

d %= 100;

int product = a \* b \* c \* d;

int last\_digits = product % 100;

if (last\_digits < 10) {

cout << 0 << last\_digits << '\n'; // append 0 to the front to make 2 digits!

}

else {

cout << last\_digits << '\n';

}

return 0;

}

```

</details>

<br>

<b>26. Problem Z</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

#include <math.h>

using namespace std;

int main() {

long long a, b, c, d ;

cin >> a >> b >> c >> d ;

// notice that the values a, b are large, so a^b will be an astronomical value

// and we cant store it in long long either, so we need to think a bit

// notice that we dont need the exact value of a^b, we just have to check

// if a^b is > c^d or not.

//

// => a^b > c^d

// => log(a^b) > log(c^d) // take log in both sides

// => b\*loga > d\*logc

// now we can do this!

if (b \* log(a) > d \* log(c)) {

// this log is in base e by default (natural logarithm (ln)),

// also the base doesn't matter here. You can also try with log2

cout << "YES\n";

}

else {

cout << "NO\n";

}

return 0;

}

```

</details>

</details>

### [Codes] Check how to solve the problems of Contest 2

<details> <summary> smash me </summary>

<b>1. Problem A</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int n; cin >> n;

for (int i = 1; i <= n; i++) {

cout << i << '\n'; // never use endl, use '\n' as it is faster than endl

}

return 0;

}

```

</details>

<br>

<b>2. Problem B</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int n; cin >> n;

if (n == 1) {

// no even numbers under 1

cout << -1 << '\n';

}

else {

for (int i = 2; i <= n; i += 2) {

cout << i << '\n';

}

// // should also work

// for (int i = 1; i <= n; i++) {

// if (i % 2 == 0) {

// cout << i << '\n';

// }

// }

}

return 0;

}

```

</details>

<br>

<b>3. Problem C</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int n; cin >> n;

int even\_numbers = 0;

int odd\_numbers = 0;

int positive\_numbers = 0;

int negative\_numbers = 0;

for (int i = 1; i <= n; i++) {

int x; cin >> x;

// taking absolute of x first before taking the modulo to get non negative remainder values

// for example: -5 % 2 = -1, 5 % 2 = 1

if (abs(x) % 2 == 0) {

even\_numbers++;

}

if (abs(x) % 2 == 1) {

odd\_numbers++;

}

if (x > 0) {

positive\_numbers++;

}

if (x < 0) {

negative\_numbers++;

}

// notice that 0 is neither positive, nor negative. It is neutral

}

cout << "Even: " << even\_numbers << '\n';

cout << "Odd: " << odd\_numbers << '\n';

cout << "Positive: " << positive\_numbers << '\n';

cout << "Negative: " << negative\_numbers << '\n';

return 0;

}

```

</details>

<br>

<b>4. Problem D</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

// use const for constant variables

// make constant variables all uppercase by convention

const int PASSWORD = 1999;

int main() {

int x;

// use while (cin >> x) to take input until End Of File (EOF)

while (cin >> x) {

if (x == PASSWORD) {

cout << "Correct\n";

break; // breaking the loop, as we dont need to take any more input as the problem said

}

else {

cout << "Wrong\n";

}

}

// note that this while loop will run until there is nothing in the file to take input from

// or when we hit the correct password

return 0;

}

```

</details>

<br>

<b>5. Problem E</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int n; cin >> n;

int maximum = 0; // set maximum to a very small value that is <= any value in the input

for (int i = 1; i <= n; i++) {

int x; cin >> x;

maximum = max(maximum, x);

}

cout << maximum << '\n';

return 0;

}

```

</details>

<br>

<b>6. Problem F</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int n; cin >> n;

for (int i = 1; i <= 12; i++) {

cout << n << " \* " << i << " = " << n \* i << '\n';

// print in the exact same format as the problem suggests

// do not print extra spaces, do not print less spaces

// 2 \* 1 = 2 vs 2\*1=2 are different!

}

return 0;

}

```

</details>

<br>

<b>7. Problem G</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int t; cin >> t;

while (t--) {

int n; cin >> n;

// use long long data type as in the worst case

// when n is the largest, that is n = 20

// n! = 20! = 2432902008176640000 which we can't store in a 32 bit int data type

long long factorial = 1;

for (int i = 1; i <= n; i++) {

factorial \*= i;

}

cout << factorial << '\n';

}

return 0;

}

```

</details>

<br>

<b>8. Problem H</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int n; cin >> n;

// prime numbers are not divisible by any number i (2 <= i < n)

for (int i = 2; i < n; i++) {

if (n % i == 0) { // check if n is divisible by i

cout << "NO\n";

// print NO and return from the program as we don't want to print NO multiple times

return 0;

}

}

// special case: 1 is not prime!

// also try to take 1 as input and run your program

// always run your code against special cases

if (n == 1) {

cout << "NO\n";

return 0;

}

cout << "YES\n";

return 0;

}

```

</details>

<br>

<b>9. Problem I</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int number; cin >> number;

int original\_number = number; // remember the number!

int reversed\_number = 0;

// get the digits from right to the left and append them one after another

// for example, for 123, get the digits from right to left -> 3, 2, 1

// then append them together: 321 (reversed number)

while (number > 0) {

int last\_digit = number % 10;

// append last\_digit to the reversed\_number

reversed\_number = reversed\_number \* 10 + last\_digit;

int number\_without\_last\_digit = number / 10;

number = number\_without\_last\_digit;

}

cout << reversed\_number << '\n';

// in the while loop we have modified the value of number

// thats why we remembered the original value

if (original\_number == reversed\_number) { // palindrome

cout << "YES\n";

}

else {

cout << "NO\n";

}

return 0;

}

```

</details>

<br>

<b>10. Problem J</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

// function to check if the number is a prime or not

// returns a boolean of true if it is a prime or false otherwise

bool is\_prime(int n) {

// always check for special cases!

if (n == 1) return false;

for (int i = 2; i < n; i++) {

// if it is divisible by some number other than 1 and n, then it is not a prime for sure

if (n % i == 0) {

return false;

}

}

// now we know that it is a prime for sure!

return true;

}

int main() {

int n; cin >> n;

// use functions to solve smaller subproblems

// this will make your code cleaner and less buggy

for (int i = 1; i <= n; i++) {

if (is\_prime(i)) {

cout << i << ' ';

}

}

cout << '\n';

return 0;

}

// Also how many operations the program is making?

// In the worst case, n = 1000

// so we are running the loop in the main function 1000 times

// and then in the is\_prime function for each number we running the loop

// 1000 times again. So in total we are making at most 1000 \* 1000 = 10^6 operations

// And in general it takes 1s to run 10^8 operations.

// So we are good and won't get TLE

```

</details>

<br>

<b>11. Problem K</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main(){

int n; cin >> n;

for (int i = 1; i <= n; i++) {

if (n % i == 0) {

cout << i << '\n';

}

}

return 0;

}

```

</details>

<br>

<b>12. Problem L</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

#include<algorithm> // for using the gcd function

using namespace std;

int main() {

int a, b; cin >> a >> b;

cout << \_\_gcd(a, b) << '\n'; // use the builtin function

return 0;

}

```

</details>

<br>

<b>13. Problem M</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

bool is\_lucky\_digit(int digit) {

// use OR operator here as if digit is 4 OR digit is 7 it returns true

// if you use AND operator here, it would mean digit would have to be 4 AND 7 at the same time!

return digit == 4 or digit == 7;

}

bool is\_lucky\_number(int n) {

// get the digits of n from right to left

// and check if there is any non lucky digit

while (n > 0) {

int last\_digit = n % 10;

if (!is\_lucky\_digit(last\_digit)) {

// if there is at least one non lucky digit then we are fucked

// return false immediately

return false;

}

int number\_without\_last\_digit = n / 10;

n = number\_without\_last\_digit;

}

return true;

}

// hey you! while reading others' codes, always start from the main function!

int main() {

int a, b; cin >> a >> b;

bool got\_lucky\_number = false;

for (int i = a; i <= b; i++) {

if (is\_lucky\_number(i)) {

cout << i << ' ';

got\_lucky\_number = true;

}

}

if (!got\_lucky\_number) {

cout << -1 << '\n';

}

return 0;

}

```

</details>

<br>

<b>14. Problem N</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

// dont forget to use meaningful variable names!

int main() {

char symbol; cin >> symbol;

int n; cin >> n;

for (int i = 1; i <= n; i++) {

int count; cin >> count;

// now print the symbol count times

for (int j = 1; j <= count; j++) { // use a different variable name in this nested loop as we have already used i in the parent loop

cout << symbol;

}

cout << '\n';

}

return 0;

}

```

</details>

<br>

<b>15. Problem O</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int row\_count; cin >> row\_count;

for (int row = 1; row <= row\_count; row++) {

// there will be "row" number of \* in the "row"th row

// for example, 1 \* in the 1st row

// 2 \* in the 2nd row and so on

int count\_of\_stars = row;

for (int i = 1; i <= count\_of\_stars; i++) {

cout << "\*";

}

cout << '\n';

}

return 0;

}

```

</details>

<br>

<b>16. Problem P</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int row\_count; cin >> row\_count;

for (int row = 1; row <= row\_count; row++) {

// there will be "row\_count - row + 1" number of \* in the "row"th row

// for example, if row\_count = 4, then 4 \* in the 1st row

// 3 \* in the 2nd row, 2 \* in the 3rd row and 1 \* in the 4th row

int count\_of\_stars = row\_count - row + 1;

for (int i = 1; i <= count\_of\_stars; i++) {

cout << "\*";

}

cout << '\n';

}

return 0;

}

```

</details>

<br>

<b>17. Problem Q</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int t; cin >> t;

while (t--) {

int n; cin >> n;

if (n == 0) {

cout << 0 << '\n';

continue;

// we can't use return 0 here as we are running multiple test cases

// and return 0 will terminate the whole program which we dont want

}

while (n > 0) {

int last\_digit = n % 10;

cout << last\_digit << ' ';

int number\_without\_last\_digit = n / 10;

n = number\_without\_last\_digit;

}

cout << '\n';

}

return 0;

}

```

</details>

<br>

<b>18. Problem R</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int n, m;

while (cin >> n >> m) {

// The program should be TERMINATED as soon as any of these two numbers

// is less than or equal to zero and don't print any thing.

if (n <= 0 or m <= 0) {

break;

}

// make n <= m for easier implementation

if (n > m) {

swap(n, m);

}

int sum = 0;

for (int i = n; i <= m; i++) {

cout << i << ' ';

sum += i;

}

cout << "sum =" << sum << '\n'; // be careful with the spaces

}

return 0;

}

```

</details>

<br>

<b>19. Problem S</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int t; cin >> t;

while (t--) {

int x, y; cin >> x >> y;

// make x <= y for easier implementation

if (x > y) {

swap(x, y);

}

int sum\_of\_odds = 0;

// loop exluding x and y

for (int i = x + 1; i < y; i++) {

if (i % 2 == 1) { // odd number

sum\_of\_odds += i;

}

}

cout << sum\_of\_odds << '\n';

}

return 0;

}

```

</details>

<br>

<b>20. Problem T</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int row\_count; cin >> row\_count;

for (int row = 1; row <= row\_count; row++) {

// there will be "2 \* row - 1" number of \* in the "row"th row

// for example, 1 \* in the 1st row

// 3 \* in the 2nd row, 5 \* in the 3rd row and 7 \* in the 4th row

int count\_of\_stars = 2 \* row - 1;

// before the stars there will be "row\_count - row" number of spaces in the "row"th row

// for example, if row\_count = 4, then 3 spaces in the 1st row

// 2 spaces in the 2nd row, 1 space in the 3rd row and 0 space in the 4th row

int count\_of\_spaces = row\_count - row;

for (int i = 1; i <= count\_of\_spaces; i++) {

cout << " ";

}

for (int i = 1; i <= count\_of\_stars; i++) {

cout << "\*";

}

// you can use the same variable names in loops if they are not nested

cout << '\n';

}

return 0;

}

```

</details>

<br>

<b>21. Problem U</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int get\_sum\_of\_digits(int n) {

int sum = 0;

// get the digits of n from right to left

// sum them up

while (n > 0) {

int last\_digit = n % 10;

sum += last\_digit;

int number\_without\_last\_digit = n / 10;

n = number\_without\_last\_digit;

}

return sum;

}

int main() {

int n, a, b; cin >> n >> a >> b;

int sum\_of\_numbers = 0;

for (int i = 1; i <= n; i++) {

int sum\_of\_digits = get\_sum\_of\_digits(i);

if (a <= sum\_of\_digits and sum\_of\_digits <= b) {

sum\_of\_numbers += i;

}

}

cout << sum\_of\_numbers << '\n';

// notice that we don't need to use long long in sum\_of\_numbers

// as in the worst case, n = 10000. And we might have to take sum of all these numbers

// from 1 to 10000. And it is at max 10000 \* (10000 + 1) / 2. And this number

// can be stored in an int variable

// Rule: Do not use long long when its not necessary

// this will make you a better coder

return 0;

}

```

</details>

<br>

<b>22. Problem V</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int n; cin >> n;

// notice that for each row the first number gets incremented by 4

// and in each row, then 3 numbers are consecutive

int first\_number\_in\_row = 1;

for (int row = 1; row <= n; row++) {

for (int i = first\_number\_in\_row; i <= first\_number\_in\_row + 2; i++) {

cout << i << ' ';

}

cout << "PUM\n";

first\_number\_in\_row += 4;

}

return 0;

}

```

</details>

<br>

<b>23. Problem W</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int n; cin >> n;

// print the first half of the diamond

for (int row = 1; row <= n; row++) {

// there will be "2 \* row - 1" number of \* in the "row"th row

// for example, 1 \* in the 1st row

// 3 \* in the 2nd row, 5 \* in the 3rd row and 7 \* in the 4th row

int count\_of\_stars = 2 \* row - 1;

// before the stars there will be "n - row" number of spaces in the "row"th row

// for example, if n = 4, then 3 spaces in the 1st row

// 2 spaces in the 2nd row, 1 space in the 3rd row and 0 space in the 4th row

int count\_of\_spaces = n - row;

for (int i = 1; i <= count\_of\_spaces; i++) {

cout << " ";

}

for (int i = 1; i <= count\_of\_stars; i++) {

cout << "\*";

}

cout << '\n';

}

// print the last half of the diamond

for (int row = 1; row <= n; row++) {

// there will be "2 \* (n - row + 1) - 1" number of \* in the "row"th row

// for example, if n = 4, 7 \* in the 1st row

// 5 \* in the 2nd row, 3 \* in the 3rd row and 2 \* in the 4th row

int count\_of\_stars = 2 \* (n - row + 1) - 1;

// before the stars there will be "row - 1" number of spaces in the "row"th row

// for example, if n = 4, then 0 space in the 1st row

// 1 space in the 2nd row, 2 spaces in the 3rd row and 3 spaces in the 4th row

int count\_of\_spaces = row - 1;

for (int i = 1; i <= count\_of\_spaces; i++) {

cout << " ";

}

for (int i = 1; i <= count\_of\_stars; i++) {

cout << "\*";

}

cout << '\n';

}

return 0;

}

```

</details>

<br>

<b>24. Problem X</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int t; cin >> t;

while (t--) {

int n; cin >> n;

int number\_of\_ones\_in\_binary = 0;

while (n > 0) {

int last\_bit = n % 2;

if (last\_bit == 1) {

number\_of\_ones\_in\_binary++;

}

int drop\_last\_bit = n / 2;

n = drop\_last\_bit;

}

int decimal = 0;

for (int i = 1; i <= number\_of\_ones\_in\_binary; i++) {

decimal = decimal \* 2 + 1;

}

cout << decimal << '\n';

}

return 0;

}

```

</details>

<br>

<b>25. Problem Y</b>

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

const int N = 45; // max value of n

int fib[N + 1]; // int data type is enough, try printing for n = 45

int main() {

int n; cin >> n;

fib[1] = 0;

fib[2] = 1;

for (int i = 3; i <= n; i++) {

fib[i] = fib[i - 1] + fib[i - 2];

}

for (int i = 1; i <= n; i++) {

cout << fib[i] << ' ';

}

cout << '\n';

return 0;

}

```

</details>

<br>

<b>26. Problem Z</b>

<details> <summary> Code(C++) </summary>

```c++

// // bruteforce (direct) solution

// #include<iostream>

// using namespace std;

// int main() {

// int k, s; cin >> k >> s;

// int count = 0;

// for (int x = 0; x <= k; x++) {

// for (int y = 0; y <= k; y++) {

// for (int z = 0; z <= k; z++) {

// if ((x + y + z) == s) {

// count++;

// }

// }

// }

// }

// cout << count << '\n';

// return 0;

// }

// // in the worst case, k = 3000

// // so we will have to run the loops 3000 \* 3000 \* 3000 = 27 \* 10^9 times

// // but in general it takes 1s to run 10^8 operations.

// // so running 27 \* 10^9 operations will take around 27 \* 10^9 / 10^8 = 270s

// // but the time limit is 3s. So it will give you TLE

// optimized solution

#include<iostream>

using namespace std;

int main() {

int k, s; cin >> k >> s;

int count = 0;

for (int x = 0; x <= k; x++) {

for (int y = 0; y <= k; y++) {

// notice that

// => x + y + z = s

// => z = s - x - y

// so if we fix x and y, then z will also be fixed

// we will just have to check if z >= 0 and z <= k or not

int z = s - x - y;

if (z >= 0 && z <= k) {

count++;

}

}

}

cout << count << '\n';

return 0;

}

```

</details>

<br>

</details>

### [Editorials and Codes] Check how to solve the problems of Contest 3

<details> <summary> smash me </summary>

<b>1. Problem A</b>

Editorial: [link](https://img.atcoder.jp/abc169/editorial.pdf)

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int a, b; cin >> a >> b;

cout << a \* b << '\n';

return 0;

}

```

</details>

<br>

<b>2. Problem B</b>

Editorial: [link](https://atcoder.jp/contests/abc209/editorial/2242)

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int a, b; cin >> a >> b;

if (a <= b) {

cout << b - a + 1 << '\n';

}

else {

cout << 0 << '\n';

}

return 0;

}

```

</details>

<br>

<b>3. Problem C</b>

Editorial: [link](https://img.atcoder.jp/abc124/editorial.pdf)

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int button\_1, button\_2;

cin >> button\_1 >> button\_2;

int ans = 0;

// press the first button twice

ans = max(ans, button\_1 + (button\_1 - 1));

// press the second button twice

ans = max(ans, button\_2 + (button\_2 - 1));

// press the first and second buttons once

ans = max(ans, button\_1 + button\_2);

cout << ans << '\n';

return 0;

}

```

</details>

<br>

<b>4. Problem D</b>

Editorial: [link](https://atcoder.jp/contests/abc226/editorial/2903)

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

#include <math.h> // for using the round function

using namespace std;

int main() {

double x; cin >> x; // we need double data type

cout << round(x) << '\n'; // use the builtin round function

return 0;

}

// Can you solve it without using a builtin function?

// Try yourself and if you can't then ask on discord

```

</details>

<br>

<b>5. Problem E</b>

Editorial: [link](https://atcoder.jp/contests/abc222/editorial/2757)

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int n; cin >> n;

// just handle seperate cases

if (n < 10) cout << "000" << n << '\n';

else if (n < 100) cout << "00" << n << '\n';

else if (n < 1000) cout << "0" << n << '\n';

else cout << n << '\n';

return 0;

}

```

</details>

<br>

<b>6. Problem F</b>

Editorial: [link](https://img.atcoder.jp/abc125/editorial.pdf)

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int a, b, t;

cin >> a >> b >> t;

int total\_biscuits = 0;

for (int time = a; time <= t; time += a) {

// produces b biscuits at this time

total\_biscuits += b;

}

cout << total\_biscuits << '\n';

// notice that the for loop runs exactly floor(t / a) times

// and each time we are adding b biscuits

// so you can solve the problem like this

// cout << (t / a) \* b << '\n';

return 0;

}

```

</details>

<br>

<b>7. Problem G</b>

Editorial: [link](https://atcoder.jp/contests/abc220/editorial/2700)

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int a, b, c; cin >> a >> b >> c;

for (int i = a; i <= b; i++) { // go through the numbers that are between a and b

// check if i is a multiple of c

// that means the remainder if we divide i by c will be 0

if (i % c == 0) {

// awesome! we just found a multiple

cout << i << '\n';

return 0; // return immediately as we don't wanna print anything else

}

}

// if we are here, it means we haven't found any multiple

// so print -1 as asked by the problem

cout << -1 << '\n';

return 0;

}

```

</details>

<br>

<b>8. Problem H</b>

Editorial: [link](https://codeforces.com/blog/entry/43392)

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int evony\_damage, ivory\_damage, damage\_goal;

cin >> evony\_damage >> ivory\_damage >> damage\_goal;

// evony and ivory can do any non-negative number of shots

// so lets try all possible shots

for (int evony\_shots = 0; ; evony\_shots++) {

if (evony\_shots \* evony\_damage > damage\_goal) {

break; // it doesn't make sense to loop further as the damage is already more than our goal

}

for (int ivory\_shots = 0; ; ivory\_shots++) {

int total\_damage = evony\_shots \* evony\_damage + ivory\_shots \* ivory\_damage;

if (total\_damage == damage\_goal) { // yayy

cout << "Yes\n";

return 0; // return immediately

}

if (total\_damage > damage\_goal) { // no need to take more shots, its already bigger

break;

}

}

}

// we are here means we haven't the damage goal

cout << "No\n";

return 0;

}

/\*\*

In our problem damage\_goal is at most 10000

The first loop will run at most damage\_goal = 10000 times as in the worst case

evony\_damage = 1 and each time he can deal 1 damage, so after 10000 loops the loop will break

But the nested inside loop will also run damage\_goal = 10000 times by the same logic

So in total, as the loops are nested, the total number of operations will be around 10000 \* 10000 = 10^8

In general 10^8 operations take around 1s or less. So we are good.

It is REALLY important to understand how many operations your problem is doing.

It is a CRIME if you don't understand this. Ask on discord if you have any questions.

\*\*/

```

</details>

<br>

<b>9. Problem I</b>

Editorial: [link](https://atcoder.jp/contests/abc199/editorial/1165)

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int a, b, c; cin >> a >> b >> c;

// for doing a^2 use a \* a instead of pow(a, 2) as pow function works for double numbers

// and it is not 100% accurate for integer calculations

if (a \* a + b \* b < c \* c) {

cout << "Yes\n";

}

else {

cout << "No\n";

}

return 0;

}

```

</details>

<br>

<b>10. Problem J</b>

Editorial: [link](https://atcoder.jp/contests/abc193/editorial/822)

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

#include <iomanip>

using namespace std;

int main() {

int regular\_price, discounted\_price;

cin >> regular\_price >> discounted\_price;

int total\_discounts = regular\_price - discounted\_price;

double discount\_percentage = (double) total\_discounts / regular\_price \* 100; // typecast to double to do double divisions

cout << fixed << setprecision(10) << discount\_percentage << '\n';

// in the problem it says "Your answer will be judged as correct when its absolute or relative error from our answer is at most 10^-2"

// so we will have to print at least 2 digits after the decimal point

// for being safe, we are printing 10 digits after the decimal point

// ALWAYS use the fixed keyword as it will make sure that you are printing exactly 10 digits after the decimal point

// without using << fixed, printing 2.63 will print 2.63, but with << fixed it will print 2.6300000000

return 0;

}

```

</details>

<br>

<b>11. Problem K</b>

Editorial: [link](https://img.atcoder.jp/abc152/editorial.pdf)

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int total\_test\_cases, passed\_test\_cases;

cin >> total\_test\_cases >> passed\_test\_cases;

if (passed\_test\_cases == total\_test\_cases) {

cout << "Yes\n";

}

else {

cout << "No\n";

}

return 0;

}

```

</details>

<br>

<b>12. Problem L</b>

Editorial: [link](https://codeforces.com/blog/entry/24160)

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

/\*\*

expression | product | last two digits

5^2 | 25 | 25

5^3 | 125 | 25

5^4 | 625 | 25

5^5 | 3125 | 25

So???

\*\*/

int main() {

long long n; cin >> n;

cout << 25 << '\n';

return 0;

}

```

</details>

<br>

<b>13. Problem M</b>

Editorial: [link](https://atcoder.jp/contests/abc214/editorial/2444)

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

int main() {

int sum, product; cin >> sum >> product;

int count = 0;

// try all possible triplets such that their sum is <= sum

for (int a = 0; a <= sum; a++) { // we start from a = 0 as they asked for triples of NON-NEGATIVE integers

for (int b = 0; b <= sum; b++) {

for (int c = 0; c <= sum; c++) {

if (a + b + c <= sum) {

// check for the product

if (a \* b \* c <= product) {

count++;

}

}

}

}

}

cout << count << '\n';

return 0;

}

// in the max case, sum = 200

// so the 3 nested loops will run sum \* sum \* sum = 200^3 = 8 \* 10^6 times

// so we are good

```

</details>

<br>

<b>14. Problem N</b>

Editorial: [link](https://codeforces.com/blog/entry/21590)

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

/\*\*

Imagine we need to find S = 1 + 2 + 3 + ... + (n - 1) + n

How to find it without using a loop?

Do you remember the formula for this from school?

If not, lets find its formula again!

S = 1 + 2 + 3 + ... + (n - 1) + n

S = n + (n - 1) + (n - 2) + ... + 2 + 1 (showing in reverse order)

----------------------------------------------------------

2S = (n + 1) + (n + 1) + (n + 1) + ... + (n + 1) + (n + 1) (sum the first and second equations)

so 2S = (n + 1) \* n (as there are n variables in total)

so S = (n + 1) \* n / 2

So instead of running a loop we can just use this formula to find the sum from 1 to n. Great!

Note that the powers of twos are 2^0 = 1, 2^1 = 2, 2^2 = 4, 2^3 = 8, 2^4 = 16, 2^5 = 32, ...

But in this problem we have to find P = -1 - 2 + 3 - 4 + 5 + 6 + 7 - 8 + 9 + 10 + ... upto n (only powers of twos have negative sign)

How to do this?

S = 1 + 2 + 3 + 4 + 5 + 6 + 5 + 8 + 9 + 10 + ...

P = -1 - 2 + 3 - 4 + 5 + 6 + 7 - 8 + 9 + 10 + ...

----------------------------------------------------------

S - P = 2 \* 1 + 2 \* 2 + 2 \* 4 + + 2 \* 8 + ... (subtracting the second equation from the first)

So basically,

S - P = 2 \* (1 + 2 + 4 + 8 + ....)

S - P = 2 \* (sum of powers of 2 that are <= n)

so P = S - 2 \* (sum of powers of 2 that are <= n)

We can compute S fast, so we have to compute the sum of powers of 2 that are <= n

But the thing is there aren't that many powers of 2 under n.

There are log2(n) powers of 2 under n. For n = 10^9, there are only 30 powers of 2 uner 10^9.

So we can just loop over them!

\*\*/

int main() {

int t; cin >> t;

while (t--) {

int n; cin >> n;

long long S = (long long) (n + 1) \* n / 2; // typecast to long long so that we don't overflow

long long sum\_of\_powers\_of\_2 = 0;

for (int i = 1; i <= n; i \*= 2) { // traversing through all powers of 2, one is just the double of the previous one

sum\_of\_powers\_of\_2 += i;

}

long long P = S - 2 \* sum\_of\_powers\_of\_2;

cout << P << '\n';

}

return 0;

}

// we can't run a loop n times in this problem thats because n can be as large as 10^9

// and in 1s we can do around 10^8 operations

// so it will take arounf 10^9 / 10^8 = 10s to run a loop n times which is way more than our time limit

```

</details>

<br>

<b>15. Problem O</b>

Editorial: [link](https://img.atcoder.jp/abc051/editorial.pdf)

<details> <summary> Code(C++) </summary>

```c++

// // bruteforce (direct) solution

// #include<iostream>

// using namespace std;

// int main() {

// int k, s; cin >> k >> s;

// int count = 0;

// for (int x = 0; x <= k; x++) {

// for (int y = 0; y <= k; y++) {

// for (int z = 0; z <= k; z++) {

// if ((x + y + z) == s) {

// count++;

// }

// }

// }

// }

// cout << count << '\n';

// return 0;

// }

// // in the worst case, k = 2500

// // so we will have to run the loops 2500 \* 2500 \* 2500 = 15625000000 times

// // but in general it takes 1s to run 10^8 operations.

// // so running 15625000000 operations will take around 15625000000 / 10^8 = 156s

// // but the time limit is 2s. So it will give you TLE

// optimized solution

#include<iostream>

using namespace std;

int main() {

int k, s; cin >> k >> s;

int count = 0;

for (int x = 0; x <= k; x++) {

for (int y = 0; y <= k; y++) {

// notice that

// => x + y + z = s

// => z = s - x - y

// so if we fix x and y, then z will also be fixed

// we will just have to check if z >= 0 and z <= k or not

int z = s - x - y;

if (z >= 0 && z <= k) {

count++;

}

}

}

cout << count << '\n';

return 0;

}

```

</details>

<br>

<b>16. Problem P</b>

Editorial: [link](https://codeforces.com/blog/entry/2393)

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

#include<string.h>

using namespace std;

const int N = 1e5 + 9;

char number\_as\_string[N];

int main() {

// as the number is at most 10^100000 so the number can have 100000 digits!

// so we can't store it in int or long long

// we have to read it as a string

cin >> number\_as\_string;

int len = strlen(number\_as\_string);

// edge case: if the length is already 1 then we don't have to perform any spell

if (len == 1) {

cout << 0 << '\n';

return 0;

}

// now we will have to convert it to its sum of digits

// but the funny thing is that the sum of digits can be stored in int

// as in the worst case every digit is 9, and sum of 100000 9s = 999999

int sum\_of\_digits = 0;

for (int i = 0; i < len; i++) {

sum\_of\_digits += number\_as\_string[i] - '0';

}

int spell\_count = 1; // we just performed one spell

// now we can perform the spells as long as we don't hit only one digit number

int current\_number = sum\_of\_digits;

while (current\_number > 9) { // it has more than one digit

int sum\_of\_digits = 0;

while (current\_number > 0) {

int last\_digit = current\_number % 10;

sum\_of\_digits += last\_digit;

int number\_without\_last\_digit = current\_number / 10;

current\_number = number\_without\_last\_digit;

}

// replace the number with its sum of digits and increment the spell count

current\_number = sum\_of\_digits;

++spell\_count;

}

cout << spell\_count << '\n';

return 0;

}

```

</details>

<br>

<b>17. Problem Q</b>

Editorial: [link](https://codeforces.com/blog/entry/610)

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

/\*\*

Divide the problem into multiple steps.

Thinking Process:

Level 1:

Loop i from 1 to n and check if i is an almost prime number,

so we can have a function for that. Lets call it is\_almost\_prime() function.

Level 2:

is\_almost\_prime(n) function:

In the function we have to check if the number n is an almost prime number.

That is we have to check if the number has exactly two prime divisors.

So we can loop over i from 1 to n and check if i is a prime divisor of n

then count it. If the count is 2 then OK.

To check if it is a divisor we can check if n % i == 0

And to check if it is a prime we need another function!

Level 3:

is\_prime(n) function:

In this function we need to check if n is a prime or not.

We can loop over i from 2 to n - 1, if any number divides it

then n is not a prime, otherwise it is a prime.

So we are done! but note that we are doing 3 nested for loops in this solution.

As n = 3000, running 3 nested loops is not good as 3000 \* 3000 \* 3000 operations will take more than our time limit (2s)

But notice that we are calling the is\_prime(n) function tons of times.

But is\_prime(n) can take 1 <= n <= 3000, so 3000 different inputs.

So we can remember which of these are prime and which are not BEFORE our solution

and we can store the answers in an array.

By doing so, we don't have to call the is\_prime(n) function in Level 3.

We can just get the info from our answer array. Check the code to understand more.

This way we can remove one nested loop.

So we will have two nested loops = 3000 \* 3000 operations which is OK

\*\*/

const int N = 3030;

bool is\_prime[N];

bool check\_prime(int n) {

if (n == 1) return false; // 1 is not a prime by definition

for (int i = 2; i < n; i++) {

if (n % i == 0) { // n is divisible by i but i is neither 1, nor n, so n must not be a prime

return false;

}

}

return true;

}

bool is\_almost\_prime(int n) { // Level 2

int prime\_divisor\_count = 0;

for (int i = 1; i <= n; i++) {

if (n % i == 0) { // i is a divisor

if (is\_prime[i]) { // prime divisor (check if it is a prime from the answer array that we precalculated!)

prime\_divisor\_count++;

}

}

}

if (prime\_divisor\_count == 2) return true; // almost prime when it has two prime divisors

else return false;

}

int main() {

int n; cin >> n;

// Level 3: precalculate which numbers are prime and which are not

for (int i = 1; i <= n; i++) {

is\_prime[i] = check\_prime(i);

}

int ans = 0;

// Level 1

for (int i = 1; i <= n; i++) {

if (is\_almost\_prime(i)) {

++ans;

}

}

cout << ans << '\n';

return 0;

}

```

</details>

<br>

<b>18. Problem R</b>

Editorial: [link](https://codeforces.com/blog/entry/6662)

<details> <summary> Code(C++) </summary>

```c++

#include<iostream>

using namespace std;

bool has\_distinct\_digits(int year) { // year has 4 digits

// get all digits from right to left

int d4 = year % 10;

year /= 10;

int d3 = year % 10;

year /= 10;

int d2 = year % 10;

year /= 10;

int d1 = year % 10;

if (d1 != d2 and d1 != d3 and d1 != d4

and d2 != d3 and d2 != d4

and d3 != d4) { // check if all digits are distinct

return true;

}

return false;

}

int main() {

int year; cin >> year;

// in this problem 1000 <= year <= 9000

// as 9012 has all distinct digits so in the following loop

// year will always have 4 digits and it will never exceed 9012

while (year++) {

if (has\_distinct\_digits(year)) {

cout << year << '\n';

return 0;

}

}

return 0;

}

```

</details>

<br>

</details>

<br>

\*\*REMEMBER THAT SOLVING MORE PROBLEMS IS THE KEY\*\*

### Similar Tutorials

- [A way to Practice Competitive Programming : From Rating 1000 to 2400+ by Masataka Yoneda](https://drive.google.com/file/d/1J2x8pIYQ3MXANgvzOgBciWd3d79j\_Exa/view)

- [From Beginner to Grandmaster - Complete Roadmap for Competitive Programming by Galen Colin](https://www.youtube.com/watch?v=bSdp2WeyuJY)

Also, remember to exercise and drink more water. It helps a lot.

Good luck <a href="https://emoji.gg/emoji/8771\_blobheart"><img src="https://emoji.gg/assets/emoji/8771\_blobheart.png" width="16px" height="16px" alt="blobheart"></a>.