CALICO Contest Guide

CALICO Spring '25

April 26th, 2025

Table of Contents

Introduction

Welcome! **The CALICO Informatics Competition** (formerly California Informatics Competition) is a biannual international programming competition hosted by students at UC Berkeley. Our mission is to promote the understanding of algorithms and help students grow their problem-solving skills in a fun and exciting way!

Alongside hosting live contests, our judge platform **CALICOJudge** hosts a problem bank where you can practice solving past contest problems any time of the day. Additionally, **our GitHub** hosts problem statements, solutions, test cases, and editorials from past problems as an excellent learning resource!

CALICO contests are run during the fall and spring seasons, and consist of a large number of problems meant to be solved in teams. Your goal during the contest is to score as many points as possible by solving problems as quickly as you can!

The rest of this guide will go through the structure, rules, and logistics of the competition. If you have any questions, please don't hesitate to email us at **team@calico.cs.berkeley.edu** or ask in our **community Discord server**!

Good luck!

Contest Format

To summarize, the contest will feature 9-13 problems spanning a wide range of topics and difficulties, each with one or more test sets. During the 3 hour contest duration, your team can make as many submissions to as many test sets of problems in any order as you'd like. Your team's goal is to solve as many test sets of problems as possible, as accurately as possible, and as quickly as possible in order to score the most points while minimizing penalty time.

Problems

Each problem consists of a problem statement and one or more test sets (see Tests section). Problem statements describe the problem that needs to be solved. They consist of multiple parts:

- **Introduction**: a story, a meme, a picture, etc. that establishes the premise/motivation of the problem. This section does not contain information needed to actually solve the problem.
- **Problem Statement**: the actual problem statement that describes the problem to be solved in clear and concise language.
- **Input/Output Format**: a definition of the input and expected output format. A program must follow this format precisely for it to be considered correct.
- **Constraints**: limits and guarantees on the types and sizes of inputs. This section is broken down into multiple subsections, as test sets may have their own different constraints.
- **Sample Test Cases**: example inputs and their expected outputs to help contestants test their understanding of the problem.

Tests

Each problem also consists of one or more test sets. Every problem has a main test set, and zero or more bonus test sets. Each test set contains a collection of input and expected output test files. Each test set is also assigned a difficulty rank and is worth a certain amount of points.

Test sets are also categorized as either main or bonus test sets. Bonus test sets are worth fewer points and are meant to reward particularly efficient solutions to problems with multiple acceptable solutions. Test sets are also almost always cumulative. This means that each bonus test set usually includes all the test files from the previous main and/or bonus test sets, along with some new ones.

Be sure to carefully read the Input/Output Format and Constraints sections of the problem statements to see what exactly will be given in the inputs and what exactly will be expected in the outputs.

Some test files are samples, while others are hidden. Sample test files can be viewed during the contest through the "Problemset" tab on CALICOJudge. Hidden tests can't be viewed during the contest.

To solve a test set, a team must make a successful submission (see Submissions section) to the test set.

Submissions

A submission is a program submitted by a team to a specific test set.

There is no limit to the number of times you can submit solutions to a problem, although it does affect scoring (see Scoring section). Your team is also not restricted to using the same programming language throughout the contest; likewise, attempts at a single problem also do not need to be in the same language.

When a submission is made, CALICOJudge runs the program against every input test file of the test set, subject to a set of restrictions (defined in the next section). The output of the program for each input is then compared against each expected output of the test set. CALICOJudge then gives one of the following verdicts to the submission based on the results of running the program:

- **CORRECT**: the output of the program matches the expected output for every input test file of the test set.
- **WRONG-ANSWER**: the output of the program does not match the expected output for some input test file of the test set.
- **TIMELIMIT**: the program took too long to run.
- **RUN-ERROR**: the program unexpectedly crashed while running.
- **COMPILER-ERROR**: the program failed to compile.

Scoring

Each test set is assigned a point value approximately based on difficulty. The sum of all test set points in a contest will always add up to 100. Teams are ranked based on the total number of points scored by solving test sets during the duration of the contest. Ties will be broken over penalty time (less is better) which is assigned as follows:

- If your team successfully solves a test set X minutes into the contest, your team accumulates X minutes of penalty time.
- Penalty time is only accumulated the first time a test set is solved.
- For every incorrect submission to a test set, your team will accumulate 10 additional minutes of penalty time. **This extra penalty will only be included if you eventually end up solving the problem.**
- Submissions that fail to compile do not accumulate penalty time.
- Problems with multiple test sets will accumulate penalty time for each test set.

Programming Languages

Contestants can make submissions in a number of different languages. The following table describes the programming languages supported by CALICOJudge.

Language	Version	Implementation	Time	Extension	Flags
C	C17	gcc 12.2.0	1x	.c	Link
C++	C++17	g++ 12.2.0	1x	.cpp, .cc, .cxx, .c++	Link
Java	Java 17	OpenJDK 17.0.14	2x	.java	Link
Python 3 (CPython)	Python 3.11.2	CPython 3.11.2	4x	.py, .py3	None
Python 3 (PyPy)	Python 3.9.16	PyPy3 7.3.11	1x	.py, .py3	None
Rust	Rust 1.63.0	Rust 1.63.0	1x	.rs	Link

Some programming languages are given a time multiplier for balancing purposes. For example, a 2x multiplier means that submissions made using that language will get double the time limit to run. This is to compensate for differing amounts of overhead between languages, and instead place emphasis on algorithmic efficiency. The last column links to the script used to compile each language's source file (if compiled), which includes compiler flags.

Program Restrictions

Your code must be smaller than 100KB in size and compile in 30 seconds or less.

Additionally, your code must also run within set time and memory limits; submissions must finish running within 1 second unless otherwise specified by the problem (subject to time limit multipliers described in the previous section). Your program must also use less than 256MB of memory unless otherwise specified by the problem. These limits are displayed underneath each problem on the CALICOJudge "Problemset" interface. Some problems may have different time and memory limits; these differences will be reflected on CALICOJudge and also in the problem statement.

Problem Ranks

Alongside being assigned different amounts of points, test sets for each problem are also sorted into four ranks, which are an approximate rating of difficulty.

- **Rank 1** problems (white) are targeted towards those who are new to coding. They use knowledge of introductory programming concepts (conditions, loops, etc.).
- **Rank 2** problems (orange) are targeted towards those with general programming and computer science experience. They make use of introductory algorithmic concepts & data structures, involving more problem-solving than Rank 1's.
- **Rank 3** problems (brown) are targeted towards those with an algorithms background and competitive programming experience. They may use algorithms or math concepts taught at UC Berkeley, and involve more logical complexity. These problems also typically require a stricter degree of computational efficiency.
- **Rank 4** problems (black) are targeted towards those with competitive programming mastery who go truly above and beyond. They are typically extremely involved, open ended, and rely on a combination of problem-solving and advanced concepts.

CALICO contests typically feature 3-4 Rank 1 problems, followed by a majority of Rank 2 and Rank 3 problems. There will only be 1-2 Rank 4 problems.

Regardless of rank, we explicitly avoid problems that boil down to the regurgitation of algorithms and libraries—we find it's a lot more fun to problem solve than recite concepts! However, knowledge in algorithms will certainly come in handy when optimizing solutions for more advanced problems.

Problem Types

There are a couple different problem types you can expect to see in a contest. These problem types determine the way your program input and output is handled. These include:

- **Standard problems**, which have a fixed input and a fixed correct output.
    - For example, finding the sum of two integers is a standard problem because there is only one correct answer.
- **Open problems**, which have a fixed input, but may have multiple correct outputs. A solution is correct if it outputs any of these.
    - For example, finding any prime number between two integers is an open problem because there may be many prime numbers within the range.
    - For another example, finding pi correct to some number of decimal places is an open problem: suppose the number of places is 5, then 3.14159, 3.1415926, 3.14159123456789, and 3.14159000000000000000000001 are all correct answers.
- **Interactive problems**, which are like a two-way conversation between your program and the judge. Both programs run at the same time; your program's outputs will be sent to the judge, whose responses will be sent back as input to your program depending on what it received (and vice versa). This continues

until the test case has either been solved or failed. Unlike standard or open problems, neither input nor output is fixed.

- For example, consider a number guessing game in which you have a certain number of attempts to figure out a secret number that has been chosen. After each guess, you're told if it's too high, too low, or correct. If you are not correct, then based on that, you can make another guess, and get another response. This continues until you either guess the right number or run out of attempts .

Contest Rules

General Guidelines

- Each team member is allowed to use their own computer with Internet access. You are free to search for resources available online.
- You are allowed to use code written by your team or code generated by tools written by your team.
- You are allowed to use code that is publicly available online prior to the start of the contest, subject to the AI policy.
- You are only allowed to use standard library functions that are included with your programming language.
- **Please be sure to also read the AI policy below.**

Prohibited Activity

- Communication with others outside your team is strictly prohibited. Do not discuss or share solutions with those outside your team.
- All code submitted must be written by members on your team or publicly available before the start of the contest or generated by tools that are publicly available before the start of the contest.
  - Templates provided by us are exempt from this rule.
  - Code generated by AI is also generally prohibited. See next section for details.
- Do not submit malicious code. This includes, but is not limited to:
  - Attempts to open network connections.
  - Attempts to slow down the judge, or create excessively large outputs.
  - Attempts to create files, or modify files/directories' permissions.
  - Attempts to run other programs and create processes.
  - Attempts to work with the operating system registry.
- Teams will be disqualified for any activity that jeopardizes or destabilizes the judging process, or engaging in other prohibited activity.
- We reserve the right to ban teams and contestants if necessary.

AI Policy

We strongly believe that there is value in learning to use AI effectively. However, with increasingly powerful problem solving and programming capabilities of recent AI systems, we believe that the continued usage of AI systems in CALICO poses a threat to contest fairness and difficulty balancing. As such, we restrict their usage.

CALICO adopts **Codeforces's AI policy** in addition to our existing rules. These rules only apply to live contests. Outside of live contests, AI tools may be used freely for practice, learning, or non-competitive problem-solving.

Keep in mind that a team will be disqualified if any team member is found guilty of cheating, so please make sure that all teammates are aware of these rules and consequences.

Contest Logistics

Registration Eligibility

The guidelines for team eligibility are as follows:

- CALICO is open to everyone! However, teams with only pre-college students are eligible to win placement prizes.
- We define "pre-college student" as anyone who has not entered college/university.
- Teams consist of up to 3 contestants.
- Teams can optionally choose to affiliate with a school or organization during registration.

Team changes (such as adding or substituting team members, moving team members around, etc.) will not be allowed after the registration deadline passes, since we will be generating accounts for the contest.

CALICOJudge Logistics

CALICO uses **CALICOJudge**, a fork of **DOMjudge** to host the contest and judge submissions. After registration closes, all contestants will be sent a registration confirmation email containing the username and password to their DOMjudge contest account. This email will contain info on how to access the contest site, instructions on how to log in, as well as other relevant information needed for the day of the contest! These accounts are separate from the self-registered accounts for the past contest/practice Problem Bank. You will not be able to login to the DOMjudge using your past problem/practice Problem Bank account.

We will post an announcement on our website once confirmation emails have been sent out. If at that point you have not received any emails from us (and are a registered team member), please contact us immediately at **team@calico.cs.berkeley.edu** with the team information (student names, emails, and organization) you registered under.

The judging interface of the contest will be identical to that of the Problem Bank, so we encourage you to set up an account there on your own beforehand to familiarize yourself with the interface.

Clarification Requests

If you are having trouble understanding a problem, or would like to clarify an aspect of a problem during the contest, you can submit a clarification request through **CALICOJudge**. Clarification requests are seen and answered by the contest organizers; responses to requests will also appear on the CALICOJudge interface. While we cannot give hints or debug code, we may be able to more precisely explain the premise to a problem.

Announcements and other notable clarifications from the contest organizers will also be published through this interface during the contest.

REMIX

To help contestants prepare for the upcoming CALICO contest and familiarize themselves with our judge platform, we run the REMIX contest: a weeklong practice contest featuring a selection of some of our favorite past problems.

Some REMIX contest info:

- 1 Week Long
- Open Internet
- Open Communication (allowed to openly discuss with other people)
- Problems in REMIX are out of order on the scoreboard, refer to problem number for intended ordering (this is a bug that we can't be bothered to fix lol)
- To compete, log in to CALICOJudge using the username and password sent in your confirmation email.

Teams will use THE SAME usernames and passwords for both the remix and the actual contest. Just as a reminder, REMIX and CALICO are TWO DIFFERENT CONTESTS.

CALICO In-Person

TODO Add a section talking about the in-person contest in the next edition of the contest guide.

Prizes

Every contest we give out lots of amazing prizes! There are a number of different ways to win prizes. The exact items and values for each prize will be announced on our website under the FAQ section, so stay tuned for updates!

Placement Prizes

The 5 highest-scoring teams will win Amazon gift cards. Historically, these cards have been valued at $500, $250, $125, $50, and $50, although they may change between contests.

Problem Bounty Prizes

We may also feature prizes as bounties for certain problems! Bounties accompany specific problems and feature additional challenges that you can pursue (alongside solving the problem itself). These are released at the beginning of the contest. Some bounties we've had before include Code Golf bounties (shortest program to solve the problem) or Speedrun bounties (first X teams to solve the problem).

Raffle Prizes

The contest features a raffle to give away additional prizes! The raffle will be run as follows:

1. During the contest, contestants earn tickets for themselves by solving problems with their team during the contest.
   - +10 tickets for the first correct submission your team makes during the contest to any test set!
   - +1 ticket for every additional correct submission from anyone on your team to any test set!
2. After the contest, contestants rank the available raffle prizes in order of preference via a form sent out after the contest.
3. Our raffle system pulls a random ticket to pick a contestant, and that contestant wins their highest ranked raffle prize that is still available.
4. If a contestant who has already won a prize through the raffle gets picked again, their ticket is ignored.
5. This process repeats until all prizes have been taken.

To ensure fairness, we'll use a program that pulls tickets using an open source raffle program written by us. The code for this program will be published after the contest, and the method for seeding the random state of the raffle program will be announced in our **Discord server**.

Technical Details

Program Inputs & Outputs

Your program should read from standard input and write to standard output. This can be done in a number of ways, but here are our recommendations:

- C: Use scanf(...) to read input and printf(...) to write output.
- C++: Use cin to read input and cout to write output.
- Java: Use the BufferedReader class (import java.io.BufferedReader) to read input and the PrintWriter class (import java.io.PrintWriter) to write output.
- Python3: Use input() to read input and print(...) to write output.

For more descriptive and specific examples of handling input and outputs in each language, check out the templates written for some past problems on **CALICOJudge**. The contest itself will also have templates available for some problems, written in a similar style. You can also always view the solutions to past problems on our GitHub.

Misc. Technical Details

Submissions should be deterministic in nature. In other words, they should always give the same output when given the same input. We periodically rejudge submissions during and after the contest, so it's important that your solutions are consistent. If a program that received an accepted verdict fails on the rejudge, you will retroactively lose points. You may still use random-number elements in your solution, but they should make use of a fixed seed.

Depending on the problem, it may not be guaranteed that integers will fit within standard 32-bit integer types. If larger data types are required (such as 64-bit integers), we may or may not make a note of this in the problem description. However, it is ultimately your responsibility to realize when these are needed.

We reserve the right to increase time limits or add/remove test cases during the contest to produce more accurate judge feedback.

All judging decisions are **final**.

Preparing for the Contest

Here's a list of things you must do/check before the contest:

- ☐ Check that you've **received credentials for CALICOJudge** after the registration deadline (and after the confirmation email announcement has been posted on the website).
- ☐ **Set up your programming environment**, making sure you've installed the approved languages you want to use. You should also have either an IDE or text editor set up as well.

Here's a list of things that we recommend you do, if you'd like:

- ☐ **Solve past problems on CALICOJudge** to familiarize yourself with the formatting of problems, as well as the judge interface.
- ☐ **Form a strategy to compete.** There are a lot of questions spanning a wide range of difficulties; forming a strategy with your team to solve problems can work wonders for performance!
- ☐ **Create a template for programs**. We already provide templates for some contest problems, but you're free to use your own!
- ☐ **Create a reference sheet** to use during the contest, reminding yourself of syntax, libraries, and (some suggestions: rounding decimals to a set number of digits, sorting strings alphabetically, etc.).

More Resources

Would you like to compete in more contests like this, or simply get more experience with competitive programming? Check out these other contests & resources to polish your programming skills:

- **USACO (USA Computing Olympiad)**: an annual online high-school programming contest, with an online training portal!
- **Lockheed Martin's Code Quest**: an annual high-school programming contest that inspired the creation of CALICO!
- **ProCo**: Stanford's annual high-school programming contest.
- **Facebook (Meta) Hacker Cup**: an annual programming contest run by Facebook, with multiple rounds you need to qualify into.
- **Croatian Open Competition in Informatics**: a monthly online contest run by the Croatian Association of Informatics.
- **Leetcode**: online library of programming problems covering a large range of algorithmic concepts, with weekly contests.
- **Codeforces**: online library of programming problems covering a large range of algorithmic concepts, with frequent contests.
- **CodeChef**: online library of programming problems covering a large range of algorithmic concepts, with frequent contests.
- **Project Euler**: online library of programming problems centered around math-based problems.