Table of Contents

Introduction

Welcome! The California Informatics Competition (CALICO) is a biannual high-school programming competition hosted at UC Berkeley. Our goal is to promote the understanding of useful algorithms and encourage students to grow their problem-solving skills! Alongside these contests, we also have an online Problem Bank where you can practice solving problems with our autograder any time of the day; their test cases, solutions, and write-ups are all available as learning resources!

CALICO contests are run during the fall and spring season, and consist of a large number of problems meant to be solved by teams. Your goal during this contest is to score as many points as possible; earn points by programming solutions to contest problems!

The rest of this guide will go through the structure, rules, and logistics of the competition. If you have any questions, don't hesitate to email us at **team@calico.berkeley.edu** or reach out to us on Instagram and Twitter **@berkeleycalico**!

Good luck!

Contest Structure

Contest Format

CALICO will consist of 10 - 25 problems spanning a wide range of difficulties; the contest lasts for 3 hours, during which you can submit as many solutions as you'd like.

Contest Submissions

Submissions to a problem will be run on a set of hidden test cases. These test cases, other than the sample test cases provided to you, will not be visible to participants. Constraints on these hidden test cases can be seen in the "Problem Constraints" section for each problem.

When the autograder runs your submission, it will compare your program's output against the problem's official output. If your code didn't exceed any restrictions (see "Program Restrictions") and the outputs match exactly, your team will be awarded full points. Otherwise, you will receive feedback from the autograder specifying what went wrong. You can always submit another solution and try again—there is no limit on the number of times you can submit solutions to a problem! Your team is not restricted to using the same language throughout the contest; likewise, attempts at a single problem also do not need to be in the same language.

Contest Scoring

Each problem is assigned a point value based on difficulty—teams will be ranked based on the number of points scored during the duration of the contest. Ties will be broken over penalty time; the team with the lowest penalty time will be ranked first.

Penalty time is assigned as follows:

- Penalty time is only accumulated when a correct submission is made to a problem.
- If you successfully complete a problem x minutes into the contest, your team accumulates x minutes of penalty time.
- For every failed submission to a problem, your team will accumulate 10 additional minutes of penalty time. This extra penalty will only be included if you end up solving the problem.
- Programs that fail to compile do not accumulate penalty time.

Approved Languages

The following programming languages are approved for use during CALICO:

- C: file extension .c
- C++: file extension .cpp, .cc, .cxx, or .c++
- Java: file extension .java
- Python3: file extension .py or .py3

Program Restrictions

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

Alongside outputting the correct solution, your code must also run within set time and memory limits; submissions must complete running within 1 second* and use up to 256MB of memory. These limits are displayed underneath each problem on the autograder "Problemset" interface. Some problems may have different time and memory limits; these differences will be reflected on the autograder site.

*Time limits are subject to the following multipliers when a given language is used for a submission:

- C: 1x
- C++: 1x
- Java: 2x
- Python3: 4x

This is to compensate for differing amounts of overhead between languages, and instead place emphasis on algorithmic efficiency.

Problem Ranks

Alongside being assigned different amounts of points, problems are sorted into four ranks (rank 1 through 4).

- **Rank 1** problems only require knowledge of introductory programming concepts (conditions, loops, etc.).
- **Rank 2** problems make use of introductory algorithmic concepts & data structures, consisting of problems more involved than Rank 1's.
- **Rank 3** problems make use of algorithmic concepts taught at Berkeley CS classes, and involve more logical complexity. These problems also typically require some degree of efficiency.
- **Rank 4** problems are extremely involved, open ended, and rely on a combination of problem-solving and advanced computing concepts.
- **Bonus Set** problems, a fifth category, are not assigned a rank; rather, they are paired with existing contest problems. They only differ in terms of the input size or introduce more logical complexity.

The majority of problems in a contest will be Rank 2 and 3, with the bulk of problems overall being Rank 2. There will only ever be at most one or two Rank 4 problems. Bonus Set problems are worth less points, meant to reward (but not require) particularly efficient solutions. Regardless of rank, we explicitly avoid problems that boil down to the regurgitation of algorithms/libraries—we find it's a lot more fun to problem solve than recite concepts! However, it never hurts to have more tools in your belt.

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, as long as the code you write is your own.
- You are allowed to use pre-written code during the contest.
- You are only allowed to use library functions that are included with your programming language.

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 your own; the use of code found on the internet is prohibited. This excludes the templates provided by us.
- Do not submit malicious code. This includes, but is not limited to:
  - Attempts to to open network connections.
  - Attempts to slow down the autograder, 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.

Contest Logistics

Registration Eligibility

The guidelines for team eligibility are as follows:

- CALICO is open to all students up to 12th grade, or of equivalent age/grade for homeschooled students.
- Each team consists of up to 3 students. One student should be designated as the team captain, in charge of registering the team.
- Students on the same team must be from the same school, with the exemption of homeschooled students.
- Teams should be affiliated with and represent the school under which they are registering.
- Established STEM programs are eligible to register teams by request. To have registrations under your program approved, reach out to use at **team@calico.berkeley.edu** first!

Your team is NOT fully registered until all team members submit signed waivers. This must be done BEFORE the deadline passes. Any team member without a signed waiver will not be able to compete.

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.

Contest Autograder Logistics

CALICO uses DOMjudge to host the contest and judge submissions. After registration closes, all participants will be sent a registration confirmation email containing the username and password to their DOMjudge contest account. These account are separate from accounts on the Problem Bank; contest accounts are not registered on the Problem Bank, and Problem Bank accounts cannot be used for the contest. 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!

Once confirmation emails have been sent out, we will post an announcement on our website saying so. 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.berkeley.edu** with the team information (student names, emails, and organization) you registered under.

The autograder setup will be almost identical to that of the Problem Bank, so we encourage you to set up an account there 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 the autograder website. Clarification requests are seen and answered by the contest organizers; responses to requests will also appear on the autograder interface. While we cannot give hints or debug code, we may be able to more precisely explain the premise to a problem.

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

Prizes

The following prizes will be awarded to the highest-scoring teams (1 per team):

- 1st Place: $200 Amazon Gift Card
- 2nd Place: $100 Amazon Gift Card
- 3rd Place: $50 Amazon Gift Card
- 4th and 5th Place: $25 Amazon Gift Card

Amazon Gift Cards will be emailed to the captains of winning teams immediately following the contest.

Technical Details

Program Inputs & Outputs

Your program will take in input and output answers via the standard input channel (also known as the console). In problem descriptions, "standard input" is referred to as STDIN, and likewise "standard output" with STDOUT.

- C: Use `scanf(...)` and `printf(...)` to read input from STDIN and output to STDOUT, respectively.
- C++: Use `cin` and `cout` to read input from STDIN and output to STDOUT, respectively.
- Java: Use the `BufferedReader` class (imported with `java.lang.BufferedReader`) and `System.out.println(...)` to read input from STDIN and output to STDOUT, respectively.
- Python3: Use `input()` and `print(...)` to read input from STDIN and output to STDOUT, respectively.

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

Misc. Technical Details

Submissions must be deterministic in nature. In other words, they must always give the same output when given the same input. We will rejudge all correct submissions after the contest, so it's important that your solutions are consistent. 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 often 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 autograder feedback.

All judging decisions are **final**.

Preparing for the Contest

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

- [ ] **Sign and submit the liability waiver** on ContestDojo—this is due BEFORE the registration deadline!
- [ ] Check that you've **received the autograder account credentials** after the registration deadline (and after the 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 problems on our Problem Bank** to familiarize yourself with the formatting of problems, as well as the autograder interface.
- [ ] **Form a strategy with your team**. There are a lot of questions spanning a wide range of difficulties; coming up with a strategy to choose/solve problems can work wonders for overall performance!
- [ ] **Create a template for programs**. We already provide templates for some problems on the contest, 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.). We generally avoid problems that involve obscure/menial tasks like those, so a reference sheet is certainly not required.

Terminology

Each problem description contains an outline of the input and output formats your program will be working with. Here are some of the terms we use to make our explanations as precise as possible:

- **Integer**: any number with no decimal or fractional component, such as -1, 0, 5, or 9001.
- **Decimal value**: any number with a decimal or fractional component, such as -1.5, 0.0, 5.92, or 9001.1009; it is usually clarified how many decimal places the value is rounded to.
- **Positive**: any number that is larger than zero; 1 is the smallest positive integer, while 0.01 is a small positive decimal value.
- **Non-positive**: any number that is not positive; in other words, any number that is less than or equal to zero.
- **Negative**: any number that is less than zero; -1 is the smallest negative integer, while -0.01 is a large decimal value.
- **Non-negative**: any number that is not negative; in other words, any number that is greater than or equal to zero.
- **Non-zero**: any number that is not zero; in other words, any integer that isn't 0 or any decimal value that isn't 0.0 (or 0.000... depending on how many decimal places the value is rounded to).
- **Values**: a blanket term that is broad in nature; usually used when multiple types of inputs or outputs are present at once.
- **List**: a collection of values, whose order doesn't typically matter.
- **Sequence**: a collection of values, whose order is typically relevant.

- **String**: a sequence of characters; it is usually clarified whether a given string includes characters other than letters.
- **Word** or **Name**: a string without any space characters (usage depends on context); it is usually clarified what other kinds of characters a name can include.
- **Characters**: a single letter or symbol; it is usually clarified whether letters, digits, or special characters are included in this set of values.
- **Letter**: any value in the English alphabet; it is usually clarified whether case is relevant or not.
- **Digit**: any integer that is 0, 1, 2, 3, 4, 5, 6, 7, 8, or 9
- **Special characters**: any character that isn't a letter or digit; this could include punctuation, spaces, symbols, quotes/parentheses, etc.; they exclude control characters unless otherwise specified.
- **Control characters**: characters with ASCII values 0 - 31; there really isn't ever a reason to use these in a problem, but you never know...

More specialized, one-off terminology may be used when appropriate to a given problem. We don't expect you to memorize all this terminology; this is mainly for us to ensure problems are written with consistent and precise wording. If any ambiguity were to arise from the wording of a problem during a contest, feel free to make a clarification request. Otherwise, if you have questions about the wording of a problem outside of a contest, feel free to email us at **team@calico.berkeley.edu**!

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 available!
- **Lockheed Martin's Code Quest**: an annual high-school programming contest that inspired the creation of CALICO!
- **ProCo**: an annual high-school programming contest run at Stanford.
- **Facebook Hacker Cup**: an annual programming contest run by Facebook, with multiple rounds you need to qualify into.
- **Google Code Jam & Google Kickstart**: Google Code Jam is run annually and consists of multiple rounds you need to qualify into, while Google Kickstart is a frequent collection of bite-size contests.
- **Croatian Open Competition in Informatics**: a monthly online high-school 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.