TCS CodeVita Season 11

FAQs, General Instructions and Best Practices

A) Frequently Asked Questions (FAQs)

Q1. I'm getting the message "User already logged in. Please try after 15 minutes". What do I do?

A1. If you're getting error "User already logged in. Please try after 15 mins", it means you have not logged out properly of your previous session. The app will give you the option to log out of all your previous sessions. Once you click that, all previous sessions will be invalidated, and you can login to a fresh session. If you're still getting an error, kindly write to us

Q2. "How to log in and take part in the contest?"
A2. Taking part in CodeVita Season 11 is a three-part activity.

1. One qualifies as per the eligibility criteria.
2. One has registered and has achieved 100% Profile completion status
3. On the day of the contest, participants can directly log in to the contest site **https://codevita.tcsapps.com** They can use their registered Email ID or Username to login, once they have entered Username or Email ID they will be prompted to enter the One-Time Code (OTC) from the Microsoft Authenticator App that they have configured during the registration for authentication. Once into the contest, the timer will start and run for the next 6 hours irrespective of personal breaks. No submissions can be made post expiry of these 6 hours.

Q3. "Can I submit solution in any language?"
A4. CodeVita supports 8 programming languages viz. C, C++, C#, Java, Perl, PHP, Python and Ruby. Extensions corresponding to their source code files are .c, .cpp, .cs, .java, .pl, .php, .py and .rb respectively. Only these submissions will be accepted by the system.

Q4. "What does attribution of code mean?"
A5. Attribution means the practice of acknowledging the original source. CodeVita is an open book contest which allows participants to refer to the internet. During the CodeVita contest if the participants have referred to any web resource, then they are expected to declare the actual web URL. This is referred to as 'Attribution of Code'

If the participant has written the entire code from scratch by oneself, then the 'Attribution of code' does not apply.

Q5. "Where do I mention the attributed code?"

A6. Before you submit your solution on CodeVita system, there will be an option which you have to select – whether you have referred to external source code. If 'Yes', then you must give the entire details of the source code. You will have to provide a complete link/URL of the code which you have referred to.

Code attribution **must be** done at submission time only. Offline mails to attach references to submitted code will not be entertained and should not be attempted also.

Q6. "What is proper attribution of Code?"

A7. Providing the entire link/URL of the source code is proper attribution of Code. Through the link/URL we should be able to access/view the lines of source code that are used in your submissions.

Generic entries like 'Google.com', 'www.codechef.com', 'stackoverflow', 'stackoverflow.com', 'Friend's Code', 'Internet', and other incomplete entries will not be accepted.

Q7. "What happens if I do not attribute the code I have referred to?"

A8. If two or more people's submission contains the same code that is referred to from an external source, the system has no way to know that, unless proper attribution is provided. This will lead to code getting tagged as plagiarized and the participant will not be considered for the next round of the contest. It is in the participant's best interest that s/he provides attribution for all code referred to from external sources during the *Submit* / *Submit (final)* actions.

Q8. "I am getting errors like 'program doesn't exist', '404 error', and sometimes my submit button is not visible."

A9. Before logging in, request you to kindly delete cached files by pressing Ctrl + Shift + Del, then ensure the checkbox for 'Cached Images and Files' is checked, and then click on 'Clear Data'. Ignoring this step might cause the erroneously cached files to misbehave on the site.

Q9. "Can I take personal breaks in between the contest?"

A10. Start the contest only once you are ready. Once the contest is started the timer will continue for the next 6 hours irrespective of personal breaks. You must manage your time accordingly. For breaks, please log out of the system by pressing the logout button otherwise there will be a waiting period of 15 minutes to log-in again.

Q10. “When I click on compile and run, its showing ‘Problem does not exists’ or “I am not able to view the ‘Upload solution’ button while submitting my code” Or “I get an error message that site cannot be reached”

A11. Please crosscheck this troubleshooting checklist:

- Ensure you are using a wired/reliable network. Mobile network usage is not recommended, as there can be network fluctuations.
- Compatible browsers are Chrome and Firefox.
- Browser being used should be updated to the latest versions.
- Ensure that JavaScript is enabled on your browser.
- Ensure that any org/institute/system level proxy is not blocking any content of CodeVita site.
- Clear the browser cache and download files, then try again.

If you use 'Online Editor' option and while trying to submit the code you find the 'Problem Does not Exist' error, please "logout" of the current session, close any online editor tabs and then try to login again.

Q11. “Since I have logged in, my clock is showing my time as 00:00:00.”

A12. CodeVita timer uses hybrid method of maintaining every participant’s time. That is, it uses the server time in combination with local system time. If your contest is not over and you think that showing above time is wrong, then often your local system time is incorrect. Please rectify the same. Please clear browser cache and files history. Then logout and login again for the new time to get reflected. You can also refer contest end time written below to timer.

Q12. “I want the answers to all the questions asked in the contest”

A13. Please note that we do not reveal the answers to the MockVita / CodeVita questions. You must successfully solve the question in the system to know the correct answer.

B) Tips on handling various status messages found in MockVita/CodeVita.

1. Compile Time Error:

A successful compilation simply returns silently. Hence your aim should be that your program is so agreeable with the compiler that the compiler happily returns silently.

If you get a CTE, do as follows.

- First and foremost, you must ensure that you use the same versions of compilers that the server-side uses. A list of compilers is provided here.
- If you are using the same compiler, then mentally you must treat Warnings as errors because warnings prevent the compilers from returning silently. So, get rid of all Warnings in your compilation process.
- If you have meticulously followed the above, it is highly unlikely that you will get CTE.
- In extremely rare cases, a negligibly small possibility exists that under heavy load, the servers and hence the compilers malfunction and hence your compilation fails. The probability of this happening is less than 0.01% because CodeVita engineering is now mature enough to handle thousands of concurrent compilations. In your thinking you should simply discount the possibility that the compiler has malfunctioned, but if you have good enough reason to suspect that this is what might have happened, then simply resubmit the code after some time. This kind of behavior is temporal and the probability of CTE vanishing on its own is high.
- Finally, whenever a CTE occurs the CodeVita Judge provides the exact error message that the compiler has produced. Using this message as a clue you should be able to troubleshoot past your compilation problems.

2. Runtime Error (RTE):

A Runtime error is caused because either your program or the runtime has thrown some exception. Since RTE could be caused because of N number of reasons it is difficult to pin-point and hence provide a crisp error message unlike CTE.

Let us divide RTE into two parts

1. Systemic Faults and
2. Submitted Program Faults.

Systemic Faults: Systemic Faults can give rise to RTE if the Judge is not properly configured to evaluate submissions. In most cases the configurations are on a per-language basis. However systemic faults exhibit binary behavior i.e., either it will work for all submissions in a given language or it may work for none of the submissions in that given language. It cannot be that some programs in a given language receive RTE and some don't. If this happens to be the case, then it is almost always Submitted Program Fault.

It is also possible that one language is configured properly, but some others aren't. Let's say that C is properly configured, and Java is not properly configured. In this case all the C submissions will be devoid of Systemic Faults whereas all the Java programs will be susceptible to Systemic Faults. Faults in configuration of one language do not have an impact on behavior of other language. It is the duty and task of CodeVita Engineering team that systemic faults are eliminated before the Judge is thrown open to CodeVita

participants. Unlike CTE, RTE is not a transient fault and does not change behavior even under load.

Submitted Program Faults: In 99.99% of cases, RTE is caused by Submitted Program Faults. Very rarely, and we do not recollect any instance in past seasons of CodeVita that a Systemic Fault causing RTE has ever been exposed in Live Rounds. Submitted Program Faults could be caused because of any reasons, but not limited to those stated below:

- Failure to adhere to input and output specifications is the number one cause of RTE.
- Known programming violations like Illegal memory access or Null Pointer Exception etc. results in RTE.
- RTE are more common to languages like C and C++ where static time type checking is lenient and hence faults manifest only at runtime.
- Any logical mistake that leads to throwing an exception receives an RTE.

If you have participated in previous seasons of CodeVita, I request you share your stories on how you got past RTEs.

3. Time Limit Exceeded (TLE):

In automated code evaluation environments optimal utilization of shared resources such as CPU and memory are key to delivering good performance. However, no matter how well a platform is engineered, performance can deteriorate if submitted code is a CPU or Memory hog. So, one poorly written program can affect the evaluation times of several others. In such situations, platforms, including CodeVita, have no choice but to abort the rogue program. The threshold when this behavior kicks in, in CodeVita, is different for different questions.

For example, if a problem is purely compute-intensive i.e., has CPU-affinity then the programs may have a smaller threshold, say 1 second. Likewise, if a program requires a lot of memory accesses to be performed, it may have a slightly higher threshold, say 2 seconds. Problem text explicitly articulates the time limit under which the program should finish in, in units of seconds. An intelligent reader will have already figured out that the moment this status message is received, one must minimize the runtime of the program.

To cite a contextual example, let's say a question in CodeVita requires you to sort millions of elements. If you implement a naive algorithm like Bubble Sort whose Order Complexity is O(n^2) you are almost certain to receive a TLE. A solution to get past TLE would be to implement a better sorting algorithm, say Quick Sort whose Order Complexity is O (n log n). This will drastically reduce the sorting time and the program can finish within thresholds.

Now that we have seen what a TLE is and why it occurs and how system responds to it, let's see some of the ways in which you can overcome TLE

- It is good practice to insert timestamps in your code to know how much time is spent in different parts of your code. So, in case you get a TLE, you already know where your bottlenecks are.
- Your choice of data structure and algorithm plays a critical role in assessing whether you will or will not receive a TLE.
- Keep a profiler handy and more importantly know how to use it so that in case of TLE you may get insights into the runtime of your code.
- With TLE, some good programmers have a reverse gripe i.e., they write so optimized code that they feel that the thresholds are too lenient. Such programmers are advised to have patience. May be with a few questions, few programmers can get away even with sub-optimal code, but it cannot happen always. Keep up the good habit of writing optimized code. There will come a question where only optimal code will pass, and the rest will receive TLE. It's just that not all questions are geared towards figuring out if the participant can write optimal code. So please don't be too hung up if you have written a (n^2) complexity algorithm and your friend's (n^3) implementation also passes the evaluation.

So, all the best to you all and let there be no more TLEs. Sophisticated programmers use many brilliant techniques, like commenting on certain sections of the code and figure out the bottlenecks if a TLE status gets converted into Wrong Answer status.

4. Memory Limit Exceeded (MLE):

Just like TLE appears due to longer than allowed runtime execution times, MLE occurs due to higher than permissible memory utilization. More memory utilization is a function of two things

- the language of your choice
- how you handle memory allocation / deallocation in your code

Memory footprint of languages: Standalone languages like C and C++ have better memory footprint than managed runtime languages like Java and C#. Interpreted languages like Perl, Python, Ruby etc. are somewhere in between. CodeVita systems are 64-bit, and we calibrate the footprint of languages before setting memory limits on them. It can be safely said for the kind of problems that are asked in CodeVita, the memory footprint of language runtimes far exceed the amount of memory that a program may possibly need to use to arrive at the correct solution. Hence MLE status is almost always due to poor memory management strategy implemented in the program.

Memory Management in own code: Statically allocating large chunks of memory or speculatively allocating memory based on own understanding of questions is usually the main reason for MLE. Other than that poor data structures and algorithms also cause more than permissible usage of memory. In CodeVita, memory footprint of every process is tracked and rogue processes using more memory are terminated and a status of MLE is returned to the submitter of that program. There are no general rules on how to reduce

memory utilization. The rules are language and context dependent. So, ensure that you are aware of how-to techniques to reduce memory utilization in language of your choice.

5. Wrong Answer:
Wrong Answer is caused because your program didn't give the same output as expected for 1 or more test cases.

Whatever questions are being asked in CodeVita goes through thorough testing. So even if your program executes successfully on your system, which does not indicate its correctness. It should pass through all the private test cases of our problem. So before raising any queries, verify your program thoroughly.

6. Accepted:
Accepted status comes when your program has passed all the test cases i.e., it provided the same output as expected.

If your program shows this status, your problem is solved, and you should move on to the next problem.

7. Presentation Error:
Presentation Error status comes when your program output differs from the expected output by a whitespace character.

Even If your program shows this status, your problem is considered as solved. So don't try to get it in Accepted status and move on to the next problem.

8. Difference between Accepted and Presentation Error:
Both Accepted and Presentation Error status are the same. If either of these two statuses come while evaluating your program, consider it as solved and move on to the next problem.

9. Public Test Case Vs Private Test Case:
While submitting solutions on CodeVita site, if using Code Editor, upon clicking on ***Compile and Run*** button, The program gets evaluated for public testcases only which are given in problem text. Getting a *Presentation Error* or *Accepted* status here only means that your program is compiling and executing successfully against public testcases. Simply executing *Compile and Run* is not enough. In order to run private testcases against programs one needs to click on the **Submit (Final)** button. This submission will be considered for Ranking. This can end up in any of the 8 statuses. Also, getting Accepted or Presentation Error while *Compile and Run* does not guarantee that the same status will appear after *Submit (Final)* functionality. As described above, this is because *Compile and Run* is run against public

testcases while *Submit (Final)* is run against private testcases. Public testcases are those testcases which are present in problem text whereas private testcases are hidden and never known to participants.

C) General Instructions and Best Practices:

- You will have 6 hours to take the contest. Your 6 hours start the moment you click on 'Start Contest' button after you login to **https://codevita.tcsapps.com** on the day of the contest.
- In order to get your full quota of 6 hours, ensure that you login to the contest at least 6 hours before the contest ends.
- If you login with less than 6 hours remaining for the overall contest, you will have less time to complete the contest.
- Verify that you can log into CodeVita platform **https://codevita.tcsapps.com** with your email ID / Username and password generated during your registration.
- Network - Ensure that you have reachability with **https://codevita.tcsapps.com** from a reliable and stable network (say home network or institute network). The mobile network is not recommended due to fluctuations in speed and quality.
- **Expert Tip**: It is advised to set up your local environment much before the contest so that you get accustomed to it. Match your language compilers and/or interpreters with the supported versions, the list can be viewed at
    - **https://codevita.tcsapps.com** -> Login -> Contest site -> Guidelines Tab - > Compilers and Interpreters sub-tab. Coding in a local System is not only fast, but also convenient as per our top coders, as it saves the code directly to your machine. Coding online invokes you to submit twice, once for public test case and then final submission for Private testcases. Compiling offline allows you to upload the final solution only once and saves time.
- Submit solutions to the questions only in supported languages like C, C++, C#, Java, Perl, PHP, Python and Ruby.
- Familiarize yourself with application controls, look and feel and feedback mechanisms.
- If you are going to be using shared infrastructure like a college Lab, ensure that **https://codevita.tcsapps.com** is accessible from your target environment. Also, check that your college firewall is not blocking the contest site.
- Site is best viewed in modern browsers like Chrome, Firefox and IE 9 and above. Ensure you are using a compatible browser.
- Ensure that your browser supports JavaScript. The site will not work properly if your browser has blocked JavaScript. Bookmark the site for quick access later.

- Please go through the guidelines, sample questions, self-help trivia and FAQs on **https://codevita.tcsapps.com** before starting the contest.
- Finally, plagiarism will have very serious consequences in the actual contest, we have an entire team and a robust system dedicated to checking plagiarism, so refrain from copying codes as a general practice.

Thank You and Happy Coding!

Best Regards,

Team CodeVita