FAQs

Want to catch a glimpse of how it's like working at Amazon?

You can refer to the software development topics on amazon.jobs.

What resource may I use during the coding challenge?

You may use the resource that are available for all candidates (e.g. JDK or STL). Use the in-browser editor to work on the coding challenge and elaborate your coding approach as much as possible. Your coding and elaboration skills are both considered in our evaluation.

Do I have to complete both scenarios in the coding challenge?

Yes, you're expected to complete both scenarios. Manage your time effectively by checking the on-screen timer regularly during the coding challenge. Don't get stuck on one question. Move on to the next one and return to it later if you need to.

Important Assessment topics

The time remaining will be clearly displayed on the screen. **You will not be able to stop the timer once you have started each test**, so we recommend that you complete each test in one sitting.

Note that efficiency and optimization, as opposed to brute force solutions, earn more points! **Your code must compile for all code questions in order to move forward in the interview process. Be sure to test your code and ensure it runs before you submit your code or before time runs out.**

You can compile your code as many times as you like during the assessment, but there must a 15 second interval between consecutive compilations.

Edge Cases: Ensure your solutions consider all edge cases and handle large inputs effectively. This is key to doing well in the assessment.

Your code is being auto-saved periodically, and you can also save it clicking on the SAVE button. In case of a system failure you can resume from the last saved instance. Your code will also auto-save when you click on NEXT QUESTION if you are going back and forth.

If you need to take a break, the best time would be *between* the Coding Test and Work Styles Assessment. If you choose to take a break, log out, and when you're ready to start, log back in with the credentials below and the test will take you where you left off last.

TECHNICAL TOPICS

Programming Languages

Familiarity with a prominent language is generally a prerequisite for success. Be familiar with the syntax and nuances of common languages – Java | Python | C# | C | C++ | Ruby | JavaScript

Data Structures

Storing and providing access to data in efficient ways. Understand the inner workings of common data structures and be able to compare and contrast their usage in various applications. Know the runtimes for common operations as well as how they use memory. Wikipedia is a great resource for brushing up on data structures.

Algorithms

Basic implementation strategies of different classes of algorithms is more important than memorizing the specific details of any given algorithm. Consider reviewing traversals and divide and conquer algorithms. Consider knowing how and when to use a breadth-first search vs. a depth-first search, and what the tradeoffs are. Knowing the runtimes, theoretical limitations, and basic implementation strategies of different classes of algorithms > memorizing specific details of any given algorithm.

Coding

The most important thing a Software Development Engineer does at Amazon is write scalable, robust, and well-tested code. Be comfortable coding by hand. Expect to be asked to write syntactically correct code—no pseudo code. Check for edge cases and validate that no bad input can slip through. The goal is to write code that's as close to production-ready as possible. This is your chance to show off your coding ability.

Object-Oriented Design

Good design is paramount to extensible, bug free, long-lived code. Using object-oriented design best practices is one way to build lasting software. Have a working knowledge of a few common and useful design patterns. Know the appropriate use of inheritance and aggregation. Expect to defend and describe your design choices.

Databases

Most of the software that we write is backed by a data store, somewhere. Many of the challenges we face arise when figuring out how to most efficiently retrieve or store data for future use. The more you know about how relational and non-relational databases work and what tradeoffs exist between them, the better prepared you will be.

Operating Systems

Be familiar with some OS topics that can affect code performance. You won't need to know how to build your own operating system from scratch, but you should have an understanding of: memory management, processes, threads, synchronization, paging, and multithreading.