Ace the Amazon Interview

Thank you for the time you've invested so far in the Amazon interview process! Below are some tips we hope you will find helpful in preparing for your meetings with our team.

Dress Code
Our dress code at Amazon is relaxed, so we encourage you to dress comfortably in business casual attire. At work, you'll find most engineers in t-shirts and jeans. No suits please.

Company Background
Know what interests you about Amazon and the team or teams you will be interviewing with. It may help to spend some time researching our specific products and features, as well as our competitors. Knowing about our product groups and how they all interact with each other will give you context around our roles and will likely prompt deeper conversations that will hopefully provide a richer interview experience. Refer back to the team link(s) I sent in my first few emails.

Event Format
You will meet 1:1 with several Amazonians. The interviewer team will be comprised of Software Development Managers and Senior Engineers. Each interview session will last approximately 45-50 minutes. These in-depth conversations will drill down on technologies you have used in your career, as well as include behaviorally oriented questions that will help us determine if there is a mutual fit for our culture. For the technical portion of your interviews, you will be whiteboard coding to solve technical problems.

Amazon uses a behavioral interview format for the non-technical aspects of this role. We will ask situational questions pertaining to your past professional experiences and are ideally looking for responses that indicate how you might perform in the future. Here's an example, "Tell me about a time a project failed." We are looking for your role in the project, what happened, your actions and the øutcome. Please bear in mind that we are looking for what YOU specifically did and NOT your team.

Amazon's Leadership Principles play an important role in our interviews and are at the root of all of our behavioral interview questions. Familiarizing yourself with our leadership principles and being able to speak about past experiences as they relate to those principals will be beneficial to your interview. I would encourage you to read through your resume, think about your major accomplishments, and come up with concrete examples of how they tie into our LPs.

Try to answer questions keeping the STAR method in mind.
S- What was the situation? What was the problem statement?
T- What was the task at hand? How did you become involved? What were you asked to do?
A- What action did you take? What challenges did you face and how did you overcome them?
R- What was the result? The outcome (in a measurable way) to the customer?

For example:
Interviewer: Give me an example of your most difficult customer interaction and how you worked through it. What was the outcome?
You: At XYZ company, I once had an internal customer who wasn't happy with the product I created. When I dug in I found that the product that I built matched the requirements laid out in the requirements building phase, which were gathered by a different team. Instead of blaming or pushing back on that team, I dug in to find out what the

customer wanted the product to do. I communicated the updated requirements to the build team and got all required approvals. We were able to build out the new features in X days. The customer was satisfied because the product was able to do what s/he needed it to do.

Technical Expectations
Amazon supports a professionally ambiguous environment; therefore, some of the questions that will be asked will be purposely vague. We are looking to see if you are willing to ask clarifying questions and push back when necessary. Do not dive straight into coding!! Remember to speak clearly and think out loud. This will help us assess your problem solving abilities.

Please be prepared to answer high level to in-depth technical questions on concepts like data structures and algorithms. These questions may be inclusive of qualifying requirements, checking edge cases, and white boarding your solutions (or using HackerRank) with our engineers. Visit www.codechef.com, www.topcoder.com or similar websites to brush up on problem solving and core Computer Science fundamentals.

Be prepared to discuss technologies listed on your resume; for example, if you list Java or C++ you should expect technical questions about your experience with these technologies. Please ask questions if you need anything clarified. Be collaborative in the interview process. We also want to learn what it would be like to work with you on a day-to-day basis in our open environment. The questions may vary from high level to a deep dive, depending on the role that you are interviewing for.

Sample Interview Topics
Below is a list of broad areas that we expect people to be familiar with. It is certainly not required that you memorize all of the information outlined below, but this should serve as a helpful reference guide for the types of things you might want to brush up on before interviewing with Amazon.

Programming Languages
We do not require that you know any specific language before interviewing for a technical position at Amazon.com, but familiarity with a prominent object oriented language is generally a prerequisite for success. Not only should you be familiar with the syntax of a language like C++, Java, or C#, but you should also know some of the language's nuances such as how memory management works, what some of the most commonly used collections or libraries are, etc. You should be able to compare languages and talk about the tradeoffs between using language X vs. language Y. Additionally, it is considered a plus to be familiar with some scripting language such as perl, ruby, awk, etc. It is also nice to know the basics of regular expression as it is now a mainstay in both the object oriented and scripting worlds.

Data Structures
Most of the work we do involves storing and providing access to data in efficient ways. This necessitates a very strong background in standard data structures. You should know what each of these data structures is and how they are implemented, what their runtimes are for common operations, and under what circumstances it would be beneficial to use a particular one. The list below is in no particular order:

- Array
- Linked List
- Tree (Tree, Binary Tree, Binary Search Tree, Red-Black Tree, etc.)
- Heap
- Hash Table
- Stack
- Queue
- Tree
- Graph (both directed and undirected)
- Algorithms

It is also important to know efficient ways to manipulate data. One great way of demonstrating this is to brush up on some common algorithms. We will expect that you can apply and discuss the tradeoffs between commonly used algorithms.

- Sorting: Bubble Sort; Merge Sort; Quick Sort; Radix/Bucket Sort
- Traversals (On multiple data structures): Depth First Search; Breadth First Search

Coding
Expect to be asked to code syntactically correct code– no pseudo code. If you are a bit rusty coding without an IDE or coding in a specific language, it is probably a good idea to dust off the cobwebs and get comfortable coding with pen and paper. The most important thing a software engineer does at Amazon.com is write scalable, stable, robust, and well tested code. These are going to be the main criteria by which your code will be evaluated, so make sure that you check for edge cases and common error inputs, as well as the "happy paths" through the code.

Object Oriented Design
Good design is paramount to extensible, bug free, and long living code. It is possible to solve software problems in almost limitless numbers of ways; but when software needs to be robust and extensible,
It is important to know some common techniques that help with this. Using object oriented design best practices is one way to build lasting software. You should have a working knowledge of a few common and useful design patterns (singleton, factory, adapter, bridge, visitor, command, proxy, observer, etc.), as well as know how to write software in an object oriented way with appropriate use of inheritance and aggregation.

Databases
Most of the software that we write is backed by a database somewhere. A lot of the challenges we face come into play when interfacing with existing data models and when designing new data models. You should know the basics of how relational databases work, how to design relational database schemas, and how to write basic SQL queries against a database.

Distributed Computing
Our systems at Amazon.com usually have to work under very strict tolerances at high load. While we have some internal tools that help us with scaling, it is important to have an understanding of a few basic distributed computing concepts. Having an understanding of topics such as map-reduce, service oriented architectures, distributed caching, and load balancing, will help you in formulating answers to some of the more complicated distributed architecture questions you might encounter.

Internet Topics
This is Amazon.com; we are an online company and we expect our engineers to be at least familiar with the basics of how the internet works. You might want to brush up on how internet browsers do what they do, DNS lookups, what TCP/IP and HTTP are, sockets, etc. We are not necessarily looking for network engineering types of qualifications, but a solid understanding of the fundamentals of how the web works is a requirement.

Operating Systems
You will not need to know how to build your own operating system, but you should be familiar with some OS topics that can affect code performance; things such as memory management, processes, threads, synchronization, paging, multithreading, deadlocks (causes, detection, avoidance).

You may be asked a technical question like this:
Interviewer: Write a function that given a string of words, returns the top ten most used words (in order).
You: "I think I could keep track of a count for each word, then sort them and return the top ten".
Interviewer: How would you store the counts?
You: "Probably a hash table"
Interview: How performant would your code be?

You: “Let’s see. We’d have to parse into words, then hash each one. Hashing should be constant time in the average case, so this will be O(n). Sorting to get the top ten would be O(n logn) if we use quicksort, so we’d get O(n logn) overall.”
Interviewer: Let’s code this up.
You: I’m using Ruby.

```
def top_words(text)
 words = text.split(' ')
 counts = {}
 words.each do |word|
  if counts.include?(word)
   counts[word] += 1
  else
   counts[word] = 1
  end
 end
 top_counts = counts.sort{ |a,b| b[1] <=> a[1] }.take(10)
 top_counts.map{|count| count.first }
end
```

Interviewer: Are there any cases you haven’t handled here?
You: Well, I don’t handle if there is punctuation. I could change my code to remove it first. And I don’t handle if there are less than ten words.
Interviewer: Anything else you could do to make this more efficient?
You: We do store the counts in a hash table, and re-sort. If we didn’t want to store those counts, we could keep a Most Frequently Used Queue and just return the first 10. That would be more expensive at run-time, but would require less storage space.

Other General Tips

<u>Do</u>

- Share details of your experience
- Be proud of your accomplishments
- Discuss challenges and mistakes, as well as what you learned from them
- Discuss your specific role in projects

<u>Do Not</u>

- Violate any NDAs you have in place
- Take credit for the work of others
- Be afraid to admit mistakes
- Use the same examples with all interviewers
- Focus on the negatives of your past job(s)

Interviewing is a two-way street and your interviewer wants you to succeed. They are not trying to trick you in any way. If you are unclear about a question, please ask for clarification.

Please let me know if you have any questions before you interview. Again thank you for your time and interest in Amazon and GOOD LUCK!