Technical Software Engineering Interview Questions

Q1. Describe the process you have for a programming task, from requirements to delivery.

The software development process or life cycle is a structure applied to the development of a software product. There are several models for such processes (such as the agile method), each describing approaches to a variety of tasks or activities that take place during the process.

1. **Requirements analysis.** Extracting the requirements of a desired software product is the first task in creating it. While customers probably believe they know what the software is to do, it may require skill and experience in software engineering to recognize incomplete, ambiguous, or contradictory requirements.
2. **Specification.** Specification is the task of precisely describing the software to be written, in a rigorous way. In practice, most successful specifications are written to understand and fine-tune applications that were already well-developed, although safety-critical software systems are often carefully specified prior to application development. Specifications are most important for external interfaces that must remain stable.
3. **Software architecture.** The architecture of a software system refers to an abstract representation of that system. Architecture is concerned with making sure the software system will meet the requirements of the product, as well as ensuring that future requirements can be addressed.

"What is a Software Development Process" (http://www.selectbs.com/analysis-and-design/what-is-a-software-development-process)

Q2. What programming languages do you use? Which three do you prefer or are most familiar with?

Interviewers expect engineers to be familiar with multiple languages. They might look for an engineer who has experience with C++ and with Java, to demonstrate the applicant has programming chops to rapidly pick up a new language. Python is a highly sought-after language. If you are applying for a full-stack role, then you should be familiar with JavaScript frameworks like React and Node.

Having some scripting experience with Perl or Python is also a big plus.

Q3. How do you implement your error handling?

Talk about writing tests, wrapping the code to catch exceptions, trying try/catch statements, and looking through the WOMM development process. Make sure that you have a well-thought-out answer to this question.

"Error Handling" (http://www.openbookproject.net/books/mi2pwjs/ch04.html)

Q4. What is the software development life cycle? What are the differences between them?

SDLC or the Software Development Life Cycle is a process that produces software with the highest quality and lowest cost in the shortest time. SDLC includes a detailed plan for how to develop, alter, maintain, and replace a software system.

SDLC involves several distinct stages, including planning, design, building, testing, and deployment. Popular SDLC models include the waterfall model, spiral model, and Agile model.

"What is SDLC? Understand the Software Development Life Cycle" (https://stackify.com/what-is-sdlc/)

Q5. What has your experience been like as part of an Agile software development process, if any?

Agile software development refers to software development methodologies centered around the idea of iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams. The ultimate value in Agile development is that it enables teams to deliver value faster, with greater quality and predictability, and greater aptitude to respond to change.

"What is Agile? What is Scrum?" (https://www.cprime.com/resources/what-is-agile-what-is-scrum/)

Q6. What is responsive design? What is the difference between fixed and fluid layouts?

1. **Responsive website design.** Websites that are built with responsive design use media queries to target breakpoints that scale images, wrap text, and adjust the layout so that the website can 'shrink to fit' any size of screen, such as the difference between desktops and mobiles.
2. **Fluid website design.** Websites that are built with fluid design use percentages as relative indicators for widths.
3. **Fixed design.** Websites that are built using fixed design rely on fixed pixel widths. While a design with fixed dimensions can sometimes be the quickest way to get up and running, it'll provide a less user-friendly experience across multiple devices.

"Responsive vs. Adaptive vs. Fluid Design" (https://learn.onemonth.com/responsive-vs-adaptive-vs-fluid-design/)

Q7. What is your process to test and find bugs in an application?

Software testing is a universally expected part of software development You need to create sets of tests and assessments to be conducted at various development stages. In fact, testing should be carried out at all stages of development, including after your main launch. Things change, platforms are updated, and errors in mobile apps that were not visible before an OS update can wreak havoc.

Usually, this means viewing the application as a whole and as their component pieces, then setting priorities in any areas that you think are more at risk than others. Tests are then conducted to confirm the functionality, and the detected defects are subsequently recorded. These defects can then be prioritized depending on their impact and severity.

“The Role of QA Management in App Development” (https://appus.software/blog/the-importance-of-quality-assurance-management-why-you-should-consider-it)

“How To Find The Most Common Bugs in Apps” (https://appus.software/blog/how-to-find-the-most-common-bugs-in-apps)

Algorithms and Data Structures Questions

Many technical questions in software engineering interviews quiz you on the fundamentals of algorithms and data structures—in order to evaluate your baseline knowledge of these vital topics. This seems like a formal process and something that’s designed to penalize people who didn’t take a formal computer science degree since most software engineers will use libraries to abstract away efficient implementations of these data structures and algorithms. However, it’s an important part of the process.

It’s important for you to understand how these data structures and algorithms actually work, especially since it will come up in interview settings where you’ll have to whiteboard your solution. This means solving the problem with a paper and pen instead of a computer. Here are a few sample questions to get you to practice.

Q1. What is a stack? What are the two basic operations of a stack?

A stack is a linear data structure with three basic operations: push (insertion of an element to the stack from the top), pop (removal of the latest element added to the stack). Some implementations of stack also allow peek, a function enabling you to see an element in a stack without modifying it. Stacks use a last-in, first-out structure – so the last element added to the stack is the first element that can be removed. Queues are a similar data structure, which work with a first-in, first-out structure. Stacks are usually implemented with an array or a linked list. You might be asked to implement a stack in an interview and to implement different operations.

“Stack Interview Questions” (https://www.interviewsortout.com/stack-interview-questions/)

Q2. Use Big O notation to describe QuickSort.

A quick sort usually works best on average cases, but there are worst-case scenarios. On average, it is O(N log N), but O(N2) in the worst case. You’ll want to use quick sort in situations where average-case performance matters a lot rather than dwelling on the worst. You’ll need to have a deep and nuanced understanding of algorithms and their performance/implementation in order to answer.

“Top Algorithms and Data Structures You Really Need To Know” (https://towardsdatascience.com/top-algorithms-and-data-structures-you-really-need-to-know-ab9a2a91c7b5)

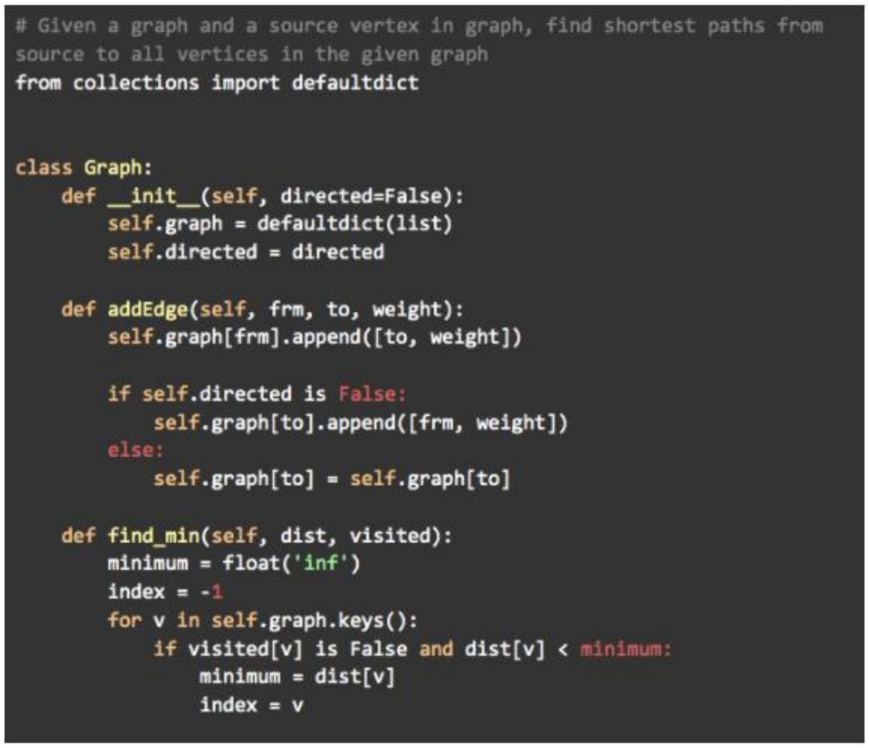
Q3. How does an array differ from a stack?

An array doesn’t have a fixed structure for how to add or retrieve data, but a stack has a strict LIFO approach (last in and first out). Questions like this will test your understanding of the nuances of data structures and the ability to memorize it.

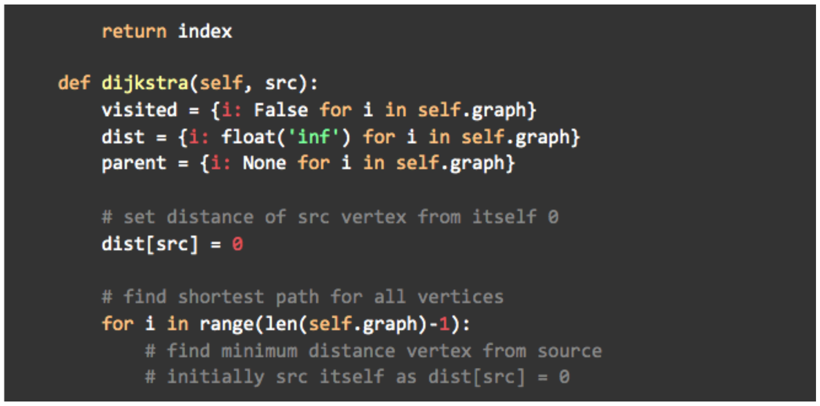
“Top Data Structure Interview Questions and Answers” (https://hackr.io/blog/data-structure-interview-questions)

Q4. Implement Dijkstra’s Shortest Path in the programming language of your choice.

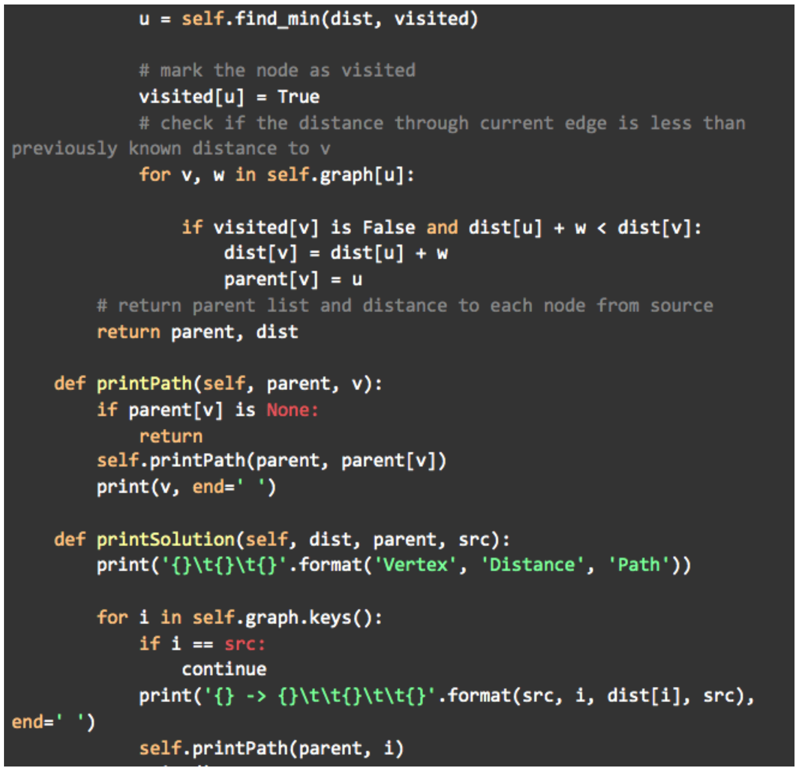
Dijkstra’s algorithm is used for finding the shortest path between nodes with positive-edge weights in a graph. This is a classic algorithm question where interviewers test your understanding of how to implement an algorithm, and you’ll often see these for more senior software development roles. Dijkstra is an example: there are others like Bellman-Ford, Floyd-Warshall. You’ll want to study different algorithms and their implementations and practice those implementations in a variety of different manners.



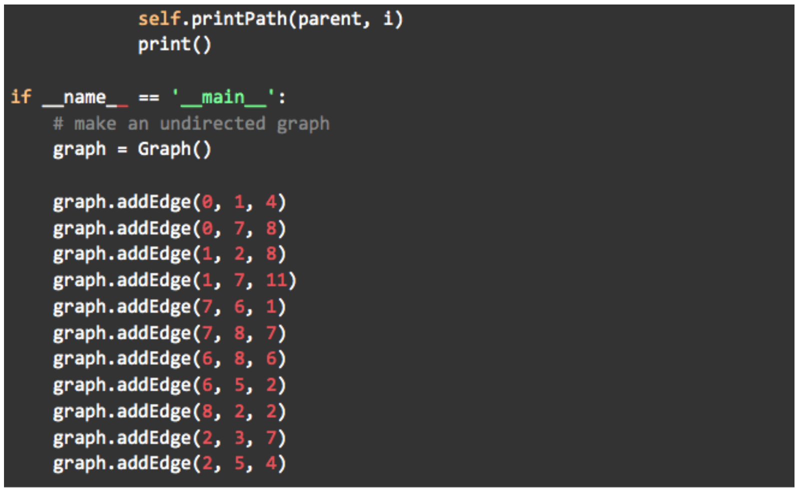
```python
# Given a graph and a source vertex in graph, find shortest paths from
source to all vertices in the given graph
from collections import defaultdict


class Graph:
    def __init__(self, directed=False):
        self.graph = defaultdict(list)
        self.directed = directed

    def addEdge(self, frm, to, weight):
        self.graph[frm].append([to, weight])

        if self.directed is False:
            self.graph[to].append([frm, weight])
        else:
            self.graph[to] = self.graph[to]

    def find_min(self, dist, visited):
        minimum = float('inf')
        index = -1
        for v in self.graph.keys():
            if visited[v] is False and dist[v] < minimum:
                minimum = dist[v]
                index = v
```



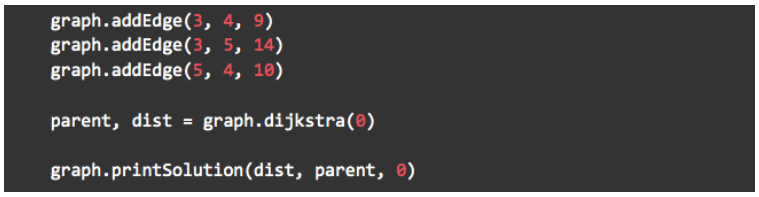
```
        return index

    def dijkstra(self, src):
        visited = {i: False for i in self.graph}
        dist = {i: float('inf') for i in self.graph}
        parent = {i: None for i in self.graph}

        # set distance of src vertex from itself 0
        dist[src] = 0

        # find shortest path for all vertices
        for i in range(len(self.graph)-1):
            # find minimum distance vertex from source
            # initially src itself as dist[src] = 0
```



```
            u = self.find_min(dist, visited)

            # mark the node as visited
            visited[u] = True
            # check if the distance through current edge is less than
previously known distance to v
            for v, w in self.graph[u]:

                if visited[v] is False and dist[u] + w < dist[v]:
                    dist[v] = dist[u] + w
                    parent[v] = u
        # return parent list and distance to each node from source
        return parent, dist

    def printPath(self, parent, v):
        if parent[v] is None:
            return
        self.printPath(parent, parent[v])
        print(v, end=' ')

    def printSolution(self, dist, parent, src):
        print('{}\t{}\t{}'.format('Vertex', 'Distance', 'Path'))

        for i in self.graph.keys():
            if i == src:
                continue
            print('{} -> {}\t\t{}\t\t{}'.format(src, i, dist[i], src),
end=' ')
            self.printPath(parent, i)
```



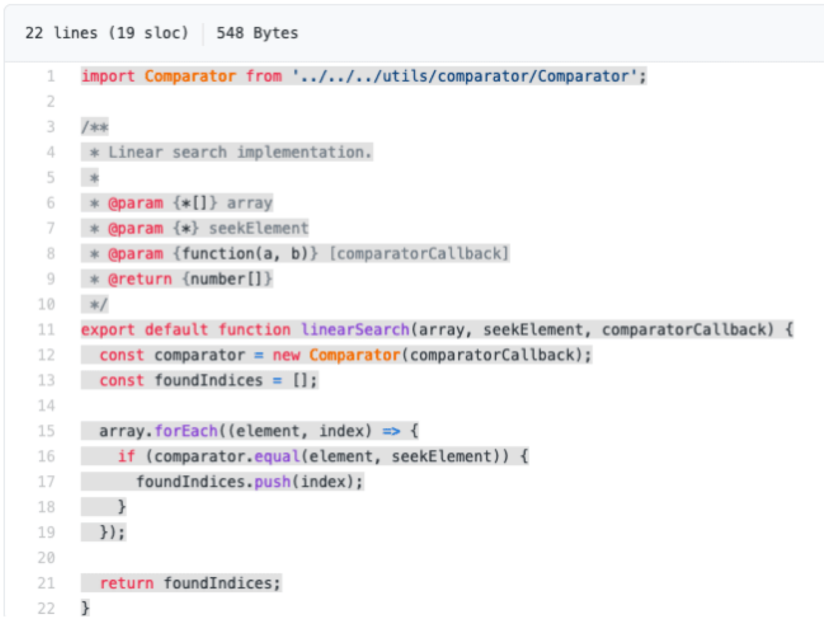
```
            self.printPath(parent, i)
            print()

if __name__ == '__main__':
    # make an undirected graph
    graph = Graph()

    graph.addEdge(0, 1, 4)
    graph.addEdge(0, 7, 8)
    graph.addEdge(1, 2, 8)
    graph.addEdge(1, 7, 11)
    graph.addEdge(7, 6, 1)
    graph.addEdge(7, 8, 7)
    graph.addEdge(6, 8, 6)
    graph.addEdge(6, 5, 2)
    graph.addEdge(8, 2, 2)
    graph.addEdge(2, 3, 7)
    graph.addEdge(2, 5, 4)
```


```
graph.addEdge(3, 4, 9)
graph.addEdge(3, 5, 14)
graph.addEdge(5, 4, 10)

parent, dist = graph.dijkstra(0)

graph.printSolution(dist, parent, 0)
```

Q5. Implement linear search in JavaScript.

This will be a test of not only your algorithm and data structure knowledge but also JavaScript knowledge and implementation. You'll want to practice in JavaScript as it's the default language for front-end web development, and you will need to know it for front-end and full-stack positions. Showing off your ability to create algorithms in JavaScript can help demonstrate this.

Linear search is a way to find a target value within a list—it checks each element in a list and sees if it matches a certain value.


```
22 lines (19 sloc) | 548 Bytes

import Comparator from '../../../utils/comparator/Comparator';

/**
 * Linear search implementation.
 *
 * @param {*[]} array
 * @param {*} seekElement
 * @param {function(a, b)} [comparatorCallback]
 * @return {number[]}
 */
export default function linearSearch(array, seekElement, comparatorCallback) {
  const comparator = new Comparator(comparatorCallback);
  const foundIndices = [];

  array.forEach((element, index) => {
    if (comparator.equal(element, seekElement)) {
      foundIndices.push(index);
    }
  });

  return foundIndices;
}
```

Quiz-style Web Developer Questions

These questions are meant more for web development positions, especially on the freelance side, rather than harder whiteboard and algorithms questions typically seen in a software development interview. See these as more of an experiential set of questions versus the theory and algorithm-based questions listed above.

Q1. What is the difference between blocking and non-blocking calls and its relationship with Node.js? Can you give an example of each?

Blocking calls are those where the execution of additional JavaScript has to wait until a non-Javascript operation (such as something with input or output) completes or finishes. You can think of this as a synchronous action. Non-blocking calls can execute asynchronously and so therefore will have a performance advantage.

This is important because JavaScript is single-threaded, which means that it executes code in a specific order and each operation must finish executing before moving onto the next operation. JavaScript has only one call stack and one memory heap. JavaScript's engine can help process asynchronous code on the browser.

Most of the I/O methods in Node.js offer a synchronous and asynchronous method. An example of a forced synchronous file read would be fs.readFileSync as a method, while the fs.readFile method would be asynchronous.

"Overview of Blocking vs. Non-Blocking" (https://nodejs.org/en/docs/guides/blocking-vs-non-blocking/)

Q2. What are web workers in HTML5, and why do they matter?

Since JavaScript is single-threaded, concurrency and simultaneous operations are difficult to execute and must be simulated with functions like setTimeout and setInterval. Web workers in HTML5 helps to run background scripts in a web application without blocking changes in the UI. In effect, web workers help simulate multi-threading in JavaScript, allowing simultaneous scripts to run.

"The Basics of Web Workers" (https://www.html5rocks.com/en/tutorials/workers/basics/)

Q3. How do you organize CSS files? What are the pros and cons of this approach?

This question tests your organizational ability and your familiarity with web development front-end principles, especially relevant if the role in question is more front-end focused.

Here's an example of a file schema for CSS that would make sense:

- **reset.css:** reset and normalization styles; minimal color, border, or font-related declarations
- **typography.css:** font faces, weights, line heights, sizes, and styles for headings and body text
- **layouts.css:** styles that manage page layouts and segments, including grids
- **forms.css:** styles for form controls and labels
- **lists.css:** list-specific styles
- **tables.css:** table-specific styles
- **carousel.css:** styles required for carousel components
- **accordion.css:** styles for accordion components."

Q4. Build a single page application with multiple sections using any framework that you feel most comfortable with

Interviewers might prefer React.js and React Router in 2020, but you can use anything you want. The purpose of this testing is to see how you build applications, even simple ones, and if you can build them at all. Oftentimes, an interviewer will observe you in a pair programming like setting, and will observe every step of your work process.

"Creating a Single-Page App in React using React Router" (https://www.kirupa.com/react/creating_single_page_app_react_using_react_router.htm)

Q5. What is black box testing? What is white box testing?

Software Testing can be majorly classified into two categories:

- **Black Box Testing** is a software testing method in which the internal structure/ design/ implementation of the item being tested is not known to the tester.
- **White Box Testing** is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.

"Software Testing Basics**" (**https://www.geeksforgeeks.org/software-testing-basics/)

"Difference between Black Box Testing vs. White Box Testing" (https://www.geeksforgeeks.org/differences-between-black-box-testing-vs-white-box-testing/)

Q6. What are some ways to make websites faster? Name as many different techniques as you can.

1. Implement your own content delivery network (CDN).
2. Use adaptive images.
3. Cache, cache, cache.
4. Evaluate your plugins.
5. Combine images into CSS sprites.
6. Enable HTTP keep-alive response headers.
7. Compress your content.
8. Configure expires headers.
9. Minify JavaScript and CSS.
10. Review your hosting package.

"10 Ways to Make Your Website Load Faster" (https://hostway.com/10-ways-to-make-your-website-load-faster/)

Q7. What is the difference between functional requirements and non-functional requirements?

Functional requirements are the features that a developed software product is expected to perform. For example, adding a payment option at an eCommerce website will be a functional requirement. Non-functional requirements measure the usability of the application such as User Interface look and feel, Security, Performance, Interoperability, Reliability, etc.

"Top 25 Software Engineering Interview Questions" (https://www.softwaretestinghelp.com/software-engineering-interview-questions/)

Q8. What is the smallest building block of ReactJS?

The smallest building blocks are React.js elements as opposed to components or props which are larger elements.

"React JS Quiz – React JS Interview Questions" (https://www.skptricks.com/2018/07/react-js-quiz-react-js-interview.html)

Q9. Why would you choose a microservice approach vs a monolithic app?

If you built your app as a microservice, it'd be a combination of different services that operate independently and robustly without being dependent on one another. You might want to do this if you wanted an app with multiple points of failures or faster performance or efficiency per each app. You should be prepared to defend your decision here and to have a point of view informed by scaling issues.

"Microservices vs Monolithic" (https://sterling.com/microservices-vs-monolithic/)

Behavioral/Culture Fit Software Engineering Interview Questions

Q1. Tell me about a tough software development problem and how you solved it.

Give a brief description. Make the assumption the other person doesn't know any specialized vocabulary or industry-specific challenges. You can also ask the interviewer about their familiarity with the topic you're about to describe and mold your answer based on the other person's level of context (a more or less technical answer).

Q2. Do you have any personal projects? Tell me about them.

Sometimes it's hard to settle on an idea for a project. If you have that problem, start by making a replica of a different application with a different tech stack or something. This will get your brain pumping and eventually you'll come up with something you'd rather do. The key isn't coming up with a great idea. The key is to get started on something.

After you've worked on your replica for a while, you might notice some shortcomings in the app that you can fix. Or you might realize that you don't want to make this replica anymore and you start on something else. The purpose of replicating an existing app isn't to really make the replica. The purpose is to get you started on something so that you'll find what you really want to do.

"Personal projects make you a better developer" (https://thenextweb.com/podium/2019/08/31/personal-projects-make-you-a-better-developer/)

Q3. Explain the concept of cloud computing to my older (not-very-technical) mother.

In the simplest terms, cloud computing means storing and accessing data and programs over the Internet instead of your computer's hard drive. Instead of storing data on your own machine, you store it on the machines of cloud service providers like Google and Amazon.

"What is Cloud Computing" (https://www.pcmag.com/news/what-is-cloud-computing)

Q4. Have you ever disagreed with your boss or manager? What did you do?

Your goal is to share a story where you disagreed with your manager and you were right about the disagreement. The reason you want to be right is that your story should ideally show how competent you are at your work, which will give the hiring manager confidence in hiring you. This answer can also display other great skills such as negotiating, selling an idea, and inspiring others.

"Answering the job interview question: Tell me about a time when you disagreed with your boss." (https://medium.com/@Boldvue/answering-the-job-interview-question-tell-me-about-a-time-when-you-disagreed-with-your-boss-f0f95b4ccb00)

Q5. Why do you want to work at [company name]? Have you used our products?

How can you help the company succeed? Read up on what's happening with the company and its industry. What stage of growth is the business in? Has it recently changed its product or service offerings? What competitive pressures is it facing? Consider this landscape and think, "What knowledge and experience do I have that would be especially useful to this employer in this time of growth and/or change?" This is where company research comes into play.

"How to Answer, Why Do You Want to Work Here?" (https://www.roberthalf.com/blog/job-interview-tips/how-to-answer-why-do-you-want-to-work-here)

Q6. When do you consider a product to be finished?

The process of software development is a never-ending cycle. The first release of a software application is rarely "finished." There are almost always additional features and bug fixes waiting to be designed, developed, and deployed.

Reports from error monitoring software about usability and bugs feedback into the process of software development and become new feature requests and improvements to existing features.

"The SDLC: 7 phases, popular models, benefits & more" (https://raygun.com/blog/software-development-life-cycle/)

Q7. Teach me about something for the next 10 minutes.

Choose a simple topic or concept that is easy to explain and will be easy for the interviewer to understand. Making the answer fun will help to engage the interviewer. Keep the answer lighthearted. Remember, the content is not as important as the delivery and showing your communication and teaching skills.

"Get Stumped During Sales Interviews" Clever Ways to Answer, Teach Me Something" (https://www.salesforcesearch.com/blog/sales-interviews-ways-answer-teach-me-something/)

Q8. What web technologies are you excited about, and why do you think they'll win/survive in the next decade?

Choose a web technology and describe it, along with reasons (for example, technical and community support) for why it might win out against other web technologies. This question tries to gauge your passion for web development and following emerging technologies, as well as your strategic vision for the future of web development.

"What Is The Future of the Web? (https://designroast.org/future-of-the-web/)

Q9. Do you contribute to open source projects? Have you flagged issues?

On this question, you'll want to flag your passion for the open-source ecosystem, as a proxy for your passion for software engineering and your ability to being proactive about contributing.

"How to Contribute to Open Source" (https://opensource.guide/how-to-contribute/)

Q10. What are your favorite resources to keep on top of software engineering?

You'll want to have a list of resources ready, but more importantly, you'll want to be pretty sharp about genuinely following resources in the space. This displays your ability to learn new things and your passion for doing so, an important trait in a field that is ever-evolving.

"Engineering Newsletters that You Should Subscribe To" (https://www.hackernoon.com/engineering-newsletters-that-you-should-subscribe-to-bd89dc9cd1c7)

Publication credit to Springboard (https://www.springboard.com/blog/21-software-engineering-interview-questions/)