TJR Bootcamp Practice Test (Sample)

Study Guide

Everything you need from our exam experts!

Table of Contents

Introduction

Preparing for a certification exam can feel overwhelming, but with the right tools, it becomes an opportunity to build confidence, sharpen your skills, and move one step closer to your goals. At Examzify, we believe that effective exam preparation isn't just about memorization, it's about understanding the material, identifying knowledge gaps, and building the test-taking strategies that lead to success.

This guide was designed to help you do exactly that.

Whether you're preparing for a licensing exam, professional certification, or entry-level qualification, this book offers structured practice to reinforce key concepts. You'll find a wide range of multiple-choice questions, each followed by clear explanations to help you understand not just the right answer, but why it's correct.

The content in this guide is based on real-world exam objectives and aligned with the types of questions and topics commonly found on official tests. It's ideal for learners who want to:

- **Practice answering questions under realistic conditions,**
- **Improve accuracy and speed,**
- **Review explanations to strengthen weak areas, and**
- **Approach the exam with greater confidence.**

We recommend using this book not as a stand-alone study tool, but alongside other resources like flashcards, textbooks, or hands-on training. For best results, we recommend working through each question, reflecting on the explanation provided, and revisiting the topics that challenge you most.

Remember: successful test preparation isn't about getting every question right the first time, it's about learning from your mistakes and improving over time. Stay focused, trust the process, and know that every page you turn brings you closer to success.

Let's begin.

How to Use This Guide

This guide is designed to help you study more effectively and approach your exam with confidence. Whether you're reviewing for the first time or doing a final refresh, here’s how to get the most out of your Examzify study guide:

1. Start with a Diagnostic Review

Skim through the questions to get a sense of what you know and what you need to focus on. Your goal is to identify knowledge gaps early.

2. Study in Short, Focused Sessions

Break your study time into manageable blocks (e.g. 30 – 45 minutes). Review a handful of questions, reflect on the explanations.

3. Learn from the Explanations

After answering a question, always read the explanation, even if you got it right. It reinforces key points, corrects misunderstandings, and teaches subtle distinctions between similar answers.

4. Track Your Progress

Use bookmarks or notes (if reading digitally) to mark difficult questions. Revisit these regularly and track improvements over time.

5. Simulate the Real Exam

Once you're comfortable, try taking a full set of questions without pausing. Set a timer and simulate test-day conditions to build confidence and time management skills.

6. Repeat and Review

Don’t just study once, repetition builds retention. Re-attempt questions after a few days and revisit explanations to reinforce learning. Pair this guide with other Examzify tools like flashcards, and digital practice tests to strengthen your preparation across formats.

There’s no single right way to study, but consistent, thoughtful effort always wins. Use this guide flexibly, adapt the tips above to fit your pace and learning style. You've got this!

Questions

1. Which approach creates an object that inherits from another?

A. Array.prototype.push

B. Math.random

C. Object.create(proto)

D. JSON.parse

2. Compare arrays and linked lists in terms of memory layout and typical operations.

A. Arrays have contiguous memory and fast random access.

B. Linked lists use arrays of pointers.

C. Arrays are dynamic in size by default.

D. Arrays have contiguous memory and fast random access; linked lists store elements in nodes with pointers, enabling cheap insertions/deletions but no constant-time random access.

3. How do you take a trade using liquidity?

A. They guarantee a trend reversal.

B. They signal the likely opposite direction and can guide entries and targets.

C. They should be ignored in all cases.

D. They indicate price will move only in the current direction.

4. Which term best describes the price level that reflects balance between supply and demand, used in this approach?

A. Liquidity sweep

B. Order block

C. Fair value gap

D. Equilibrium price

5. What is the worst-case space complexity of breadth-first search on a graph in terms of vertices V and edges E?

A. O(V + E)

B. O(V)

C. O(E)

D. O(1)

6. What is the general space-time tradeoff when using indexing in a database?

A. Indexing speeds reads at the cost of extra storage and slower writes.

B. Indexing speeds writes but increases storage.

C. Indexing has no effect on performance.

D. Indexing reduces storage.

7. Which design pattern lets observers subscribe to changes in another object?

A. Singleton pattern

B. Factory pattern

C. Observer pattern

D. Adapter pattern

8. Differentiate between amortized and worst-case time complexity with an example.

A. Amortized time complexity considers a single operation; worst-case averages across a sequence.

B. Amortized averages over a sequence of operations; worst-case is the maximum for a single operation.

C. Amortized is always better than worst-case.

D. Worst-case is about average.

9. What is a binary search tree and what is its key property?

A. A BST is a balanced tree where every node has the same number of children.

B. A BST stores keys in random order for fast insertion.

C. A BST stores keys in sorted order where left subtree keys are less than the node and right subtree keys are greater, enabling efficient search, insert, and delete.

D. A BST is a graph with cycles.

10. Which of the following is a common technique used in responsive design?

A. Fluid grids with relative units and media queries

B. Absolute positioning

C. Fixed pixel grids

D. Inline styles

Answers

1. C
2. D
3. B
4. D
5. A
6. A
7. C
8. B
9. C
10. A

Explanations

1. Which approach creates an object that inherits from another?

A. Array.prototype.push

B. Math.random

C. Object.create(proto)

D. JSON.parse

Prototypal inheritance is about linking objects so one can read properties from another through a prototype chain. Using Object.create(proto) makes a new object whose internal prototype is the object you pass in, so the new object automatically inherits properties and methods from that object without running a constructor. This is the direct way to establish inheritance between two objects in JavaScript. For example, const parent = { sayHi() { console.log('hi'); } }; const child = Object.create(parent); child.sayHi(); // inherited from parent Other common operations don't create such a link. Pushing onto an array just adds an element to that array. Math.random returns a primitive number, not an object with inheritance. JSON.parse turns text into a plain object or array with its default prototype, and it doesn't establish inheritance from another object unless you manually adjust its prototype after parsing.

2. Compare arrays and linked lists in terms of memory layout and typical operations.

A. Arrays have contiguous memory and fast random access.

B. Linked lists use arrays of pointers.

C. Arrays are dynamic in size by default.

D. Arrays have contiguous memory and fast random access; linked lists store elements in nodes with pointers, enabling cheap insertions/deletions but no constant-time random access.

Memory layout dictates how we access and update data for these structures. Arrays allocate a single, contiguous block of memory, so every element sits next to the others. This layout lets you compute any element's addressFrom its index in constant time, giving fast random access. But because the elements are packed together, inserting or removing in the middle usually means shifting a bunch of elements, which can be costly. Linked lists store each value in its own node and connect nodes with pointers. The elements aren't stored contiguously, so you must follow pointers to reach positions, which means random access isn't constant-time. However, if you already have a reference to the position where you want to insert or delete, you can adjust a few pointers and perform the operation cheaply, without moving many other elements. Additional memory overhead comes from storing the pointers in each node. So the statement that arrays have contiguous memory and fast random access, while linked lists use nodes with pointers to enable cheap insertions and deletions but lack constant-time random access, captures the typical trade-offs accurately.

3. How do you take a trade using liquidity?

A. They guarantee a trend reversal.

B. They signal the likely opposite direction and can guide entries and targets.

C. They should be ignored in all cases.

D. They indicate price will move only in the current direction.

Liquidity zones are areas where large resting orders sit, so price naturally moves toward them to fill those orders. As price approaches these pockets, the order flow can cause a pause or reversal, creating a setup you can use to time entries and set targets. By watching how price reacts at a liquidity zone—whether it shows rejection, a consolidation, or a breakout—you get a probabilistic sense of the next move. If you see a convincing setup near a liquidity area, you can enter with the expectation that the next leg will unfold toward the subsequent liquidity level or major top/bottom, setting your target accordingly. This approach helps you align entries and risks with where the market is likely to move next, without promising a reversal every time.

4. Which term best describes the price level that reflects balance between supply and demand, used in this approach?

A. Liquidity sweep

B. Order block

C. Fair value gap

D. Equilibrium price

Equilibrium price is the price level where supply equals demand, and the market balances. At this point, the quantity buyers want to purchase matches the quantity sellers want to offer, so there's no inherent pressure for the price to move up or down. If demand exceeds supply, the price tends to rise toward the equilibrium; if supply exceeds demand, the price tends to fall toward it. This price acts as the market-clearing point, reflecting the overall balance between how much is available and how much buyers want. Other terms describe more specific trading concepts focused on liquidity, blocks of orders, or price gaps, rather than the broad balance point of the entire market. They don't capture the fundamental notion of a market-clearing price, which is why the equilibrium price is the best fit here.

5. What is the worst-case space complexity of breadth-first search on a graph in terms of vertices V and edges E?

A. O(V + E)

B. O(V)

C. O(E)

D. O(1)

BFS needs to hold the graph structure plus the data it uses to track progress. It processes vertices level by level using a queue and a visited set, which consume space that scales with the number of vertices. At the same time, the graph itself—whether stored as adjacency lists or a matrix—uses space proportional to both vertices and edges, specifically O(V + E). In the worst case, you're keeping both the graph representation and the BFS data structures in memory, so the total space requirement is O(V + E). That's why this option is the best choice. Using only O(V) would miss the edge component, since the graph's edges contribute to memory usage. O(E) would miss the vertices. O(1) isn't possible because you must store at least the frontier and the visited markers.

6. What is the general space-time tradeoff when using indexing in a database?

A. Indexing speeds reads at the cost of extra storage and slower writes.

B. Indexing speeds writes but increases storage.

C. Indexing has no effect on performance.

D. Indexing reduces storage.

Indexes speed up reads by letting the database locate data quickly through a separate data structure, but they require extra storage and add overhead to write operations because the index must be updated whenever the underlying data changes. This creates a tradeoff: faster read performance at the cost of more storage usage and slower writes. That's why the best statement is that reads are sped up at the cost of extra storage and slower writes.

7. Which design pattern lets observers subscribe to changes in another object?

A. Singleton pattern

B. Factory pattern

C. Observer pattern

D. Adapter pattern

The key idea here is that one object can notify others about its changes through a subscription mechanism. In this pattern, a subject keeps a list of observers that have expressed interest, and whenever the subject's state changes, it sends out a notification to all registered observers. Observers can subscribe or unsubscribe at runtime, and their responses are triggered by those notifications. This decouples the subject from the observers, so the subject doesn't need to know who's watching or what they'll do with the update. That dynamic, broadcast-style relationship is what makes this pattern the right fit for letting observers subscribe to changes in another object. A practical example is a model in a UI: when the model changes, it notifies all registered views, and each view updates accordingly. The other patterns aren't about notifying dependents: a Singleton ensures a single instance, a Factory focuses on creating objects, and an Adapter translates interfaces between components.

8. Differentiate between amortized and worst-case time complexity with an example.

A. Amortized time complexity considers a single operation; worst-case averages across a sequence.

B. Amortized averages over a sequence of operations; worst-case is the maximum for a single operation.

C. Amortized is always better than worst-case.

D. Worst-case is about average.

The key idea here is how we measure time costs across operations versus a single operation. Amortized time complexity looks at the average cost per operation over a sequence of operations, spreading out expensive steps across many cheap ones so the typical cost remains small. Worst-case time complexity, on the other hand, cares about the maximum time any single operation could take, regardless of the sequence. A classic example is a dynamic array that doubles in size when it runs out of space. Most appends cost O(1) because you just place the element in the next available slot. Occasionally, you must resize, which involves allocating a bigger array and copying all elements, an O(n) operation. If you perform many appends, the total time grows linearly with the number of appends, so the average cost per append—the amortized cost—is O(1). However, the worst-case cost for a single append can still be O(n) during a resize. That aligns with the statement that amortized averages over a sequence of operations, while worst-case is the maximum for a single operation. The other choices miss this distinction: amortized is not about a single operation, worst-case is not about the average, and amortized is not guaranteed to be better in every situation.

9. What is a binary search tree and what is its key property?

A. A BST is a balanced tree where every node has the same number of children.

B. A BST stores keys in random order for fast insertion.

C. A BST stores keys in sorted order where left subtree keys are less than the node and right subtree keys are greater, enabling efficient search, insert, and delete.

D. A BST is a graph with cycles.

The main idea being tested is how a binary search tree uses a specific ordering of keys to enable fast lookups and updates. In a binary search tree, each node stores a key, and all keys in the left subtree are strictly smaller than the node's key, while all keys in the right subtree are strictly greater. This invariant lets you locate any target by comparing it to the current node: if the target is equal, you're done; if it's smaller, you move to the left subtree; if larger, you move to the right subtree. Because each comparison narrows the search space, operations like search, insert, and delete run efficiently on average—logarithmic time in a balanced tree. If the tree becomes unbalanced, performance can drop toward linear time, which is why balancing schemes exist. Also, an in-order traversal of a BST yields keys in sorted order, reflecting how the structure organizes data.

10. Which of the following is a common technique used in responsive design?

A. Fluid grids with relative units and media queries

B. Absolute positioning

C. Fixed pixel grids

D. Inline styles

Responsive design rests on building layouts with fluid grids and using media queries to adapt at different screen sizes. A fluid grid sizes elements with relative units like percentages rather than fixed pixels, so the layout can stretch or shrink smoothly as the viewport changes. Media queries then apply different CSS rules at chosen breakpoints, allowing the same page to reflow—columns can stack, font sizes can scale, and images can resize—so the content remains usable on phones, tablets, and desktops. This approach keeps a single, flexible layout that fits varied devices. In contrast, absolute positioning fixes elements in exact spots, which can cause overlap or content being pushed off-screen on smaller screens. Fixed pixel grids lock widths to exact pixels, failing to scale and often requiring horizontal scrolling or awkward spacing as the viewport changes. Inline styles are inflexible for responsive behavior because they hard-code styles on elements and are hard to override with media queries, making it hard to maintain a truly adaptable design.

Next Steps

Congratulations on reaching the final section of this guide. You've taken a meaningful step toward passing your certification exam and advancing your career.

As you continue preparing, remember that consistent practice, review, and self-reflection are key to success. Make time to revisit difficult topics, simulate exam conditions, and track your progress along the way.

If you need help, have suggestions, or want to share feedback, we'd love to hear from you. Reach out to our team at hello@examzify.com.

Or visit your dedicated course page for more study tools and resources:

https://tjrbootcamp.examzify.com

We wish you the very best on your exam journey. You've got this!