YEOMAN (YN) PRACTICE TEST (SAMPLE)

STUDY GUIDE

SAMPLE STUDY GUIDE WITH LIMITED QUESTIONS

VISIT US ONLINE FOR THE FULL VERSION STUDY GUIDE

https://yeoman.examzify.com

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 methods would you use to read and modify package.json within a generator?

A. this.fs.readJSON and this.fs.extendJSON

B. this.fs.readJSON and this.fs.writeJSON

C. this.fs.openJSON and this.fs.updateJSON

D. this.fs.fetchJSON and this.fs.mergeJSON

2. Which NEC is most appropriate for Submarine Nuclear Supervisor Electrician's Mate?

A. 3354

B. 3364

C. 3383

D. 3393

3. Why would you use composition over copying code into a single generator?

A. To avoid dependency management

B. To reuse existing generators and build complex scaffolds from modular parts

C. To reduce code readability

D. To force a single-file generator

4. Which instruction provides authority and procedures for the Records of Disposition Program?

A. SECNAVINST 5215.5C

B. SECNAVINST 5210.1

C. DoDI 5015.02

D. Navy Procedures Manual

5. General Aircraft Records use which SSIC?

A. 13100

B. 8080

C. 6470

D. 10120

6. Which of the following is the correct method to copy a binary file without applying templates in Yeoman?

A. this.fs.copyTpl

B. this.fs.copyJSON

C. this.fs.write

D. this.fs.copy

7. How can you validate user input in a Yeoman prompt?

A. Provide a validate function in each prompt object.

B. Use a shared validator utility.

C. Validation is not supported in Yeoman prompts.

D. Validate by checking answers after generation.

8. How can you ensure a generator is installable by npm users without pulling in heavy dependencies?

A. Use minimal runtime dependencies and peerDependencies/optionalDependencies; avoid bundling heavy modules; lazy-load as needed

B. Bundle all dependencies into the package

C. Require heavy dependencies for all setups

D. Limit installation to development environments

9. How can generator authors reduce cross-platform template path issues?

A. Use this.templatePath and this.destinationPath with path.join to build paths

B. Hard-code paths to known directories

C. Avoid using path module altogether

D. Use absolute paths for all platforms

10. Which elements are used to file records?

A. Subject ID number; Consecutive number; Issuing authority

B. Subject Name; Date of birth; Issuing authority

C. Subject ID number; Date of birth; File number

D. Subject ID number; Consecutive number; Receiving authority

Answers

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

Explanations

1. Which methods would you use to read and modify package.json within a generator?

A. this.fs.readJSON and this.fs.extendJSON

B. this.fs.readJSON and this.fs.writeJSON

C. this.fs.openJSON and this.fs.updateJSON

D. this.fs.fetchJSON and this.fs.mergeJSON

Reading and updating a JSON file in a Yeoman generator is done by first loading the current content with a readJSON call, which parses package.json into a JavaScript object you can inspect and modify. To apply changes without overwriting what's already there, use extendJSON to merge your updates into the existing file and then write it back. This combination lets you safely add dependencies, scripts, or other fields while preserving the rest of the file's content and structure. The other options either aren't part of Yeoman's file system API or would replace the entire file rather than merging changes, which isn't ideal when modifying package.json.

2. Which NEC is most appropriate for Submarine Nuclear Supervisor Electrician's Mate?

A. 3354

B. 3364

C. 3383

D. 3393

Understanding the right NEC here means identifying the one that explicitly designates the supervisory role within a submarine's nuclear propulsion electrical work. The Submarine Nuclear Supervisor Electrician's Mate NEC is crafted for Electrician's Mates who supervise and lead electrical tasks on a submarine's nuclear plant—overseeing safe operation, maintenance, procedures, and often mentor junior sailors. That combination of nuclear submarine responsibilities and supervisory authority is what makes this NEC the correct fit. The other options represent different, non-supervisory or non-nuclear/submarine roles, so they don't match the specific duties implied by "Submarine Nuclear Supervisor Electrician's Mate."

3. Why would you use composition over copying code into a single generator?

A. To avoid dependency management

B. To reuse existing generators and build complex scaffolds from modular parts

C. To reduce code readability

D. To force a single-file generator

Composition in Yeoman is about calling other generators from within your generator. This lets you assemble a project by combining small, focused generators instead of copying code into one big file. You gain reuse of proven functionality, easier maintenance because a fix in a shared generator benefits all projects that compose it, and the ability to mix and match pieces to create a tailored scaffold. For example, you could compose with a frontend generator and a backend API generator so you end up with a consistent structure, dependencies, and configuration without duplicating logic. It supports passing options and prompts, and keeps concerns separated while still delivering a single end-to-end scaffold when run. Avoiding dependency management isn't the aim; composition relies on integrating other generators. It also doesn't aim to reduce readability or force a single file; it encourages modular, readable, multi-generator setups.

4. Which instruction provides authority and procedures for the Records of Disposition Program?

A. SECNAVINST 5215.5C

B. SECNAVINST 5210.1

C. DoDI 5015.02

D. Navy Procedures Manual

Authority to run the Records of Disposition Program within the Navy is defined in the SECNAV Instruction that governs the Navy's Records Management Program. This instruction explicitly establishes responsibilities, policy, and step-by-step procedures for how records are scheduled, appraised for retention, and disposed of or transferred to archives. It provides the precise rules and processes you'd follow to manage the lifecycle of records, ensuring compliance with federal requirements and DoD guidance. While there is broader DoD guidance on records management, the Navy-specific instruction is the formal, authoritative source for how the Records of Disposition Program is implemented in the Navy. A generic Navy Procedures Manual does not carry the same formal authority for this program, and other options address different topics.

5. General Aircraft Records use which SSIC?

A. 13100

B. 8080

C. 6470

D. 10120

SSIC codes are used to categorize records by subject so you can file and retrieve them consistently. For aviation-related materials, there is a specific header in the SSIC taxonomy dedicated to General Aircraft Records, which is the designated code used for these items. Using that header groups the aircraft maintenance logs, flight records, and related documents together, making retention, search, and management straightforward. The other codes point to different subjects, so they wouldn't correctly classify General Aircraft Records. Therefore, the code assigned for General Aircraft Records is the best, most precise choice for organizing these records.

6. Which of the following is the correct method to copy a binary file without applying templates in Yeoman?

A. this.fs.copyTpl

B. this.fs.copyJSON

C. this.fs.write

D. this.fs.copy

When you need to duplicate a binary file in a Yeoman generator without applying any template processing, you should copy the file as-is using this.fs.copy. The copy operation transfers the file's bytes without interpreting or replacing placeholders, so the binary data remains intact. In contrast, the template-aware method (copyTpl) would try to render EJS-like templates inside the file, which can corrupt binary data. Writing manually would require you to supply the exact content, which isn't appropriate for a direct binary copy. Therefore, this.fs.copy is the correct approach.

7. How can you validate user input in a Yeoman prompt?

A. Provide a validate function in each prompt object.

B. Use a shared validator utility.

C. Validation is not supported in Yeoman prompts.

D. Validate by checking answers after generation.

In Yeoman prompts, you validate input by providing a validate function on each prompt object. This function runs as the user enters their answer and must return true when the input is valid or return a string with an error message to show feedback and block advancing. This per-prompt validation lets you tailor rules for every piece of input—for example, ensuring a non-empty project name, matching a specific pattern, or checking numeric ranges—before the generator proceeds. If you wanted to reuse the same checks, you could extract them into a shared utility, but the standard and most reliable approach is to attach a validate function to each prompt. Validating after generation is not ideal because it won't prevent the creation of invalid data or files in the first place.

8. How can you ensure a generator is installable by npm users without pulling in heavy dependencies?

A. Use minimal runtime dependencies and peerDependencies/optionalDependencies; avoid bundling heavy modules; lazy-load as needed

B. Bundle all dependencies into the package

C. Require heavy dependencies for all setups

D. Limit installation to development environments

The main idea is to keep a package lightweight at install time so npm users can install it without pulling in heavy dependencies. To achieve this, avoid shipping large modules as runtime requirements. Instead, declare what is truly required from the host environment and load heavier code only when it's actually needed. Using minimal runtime dependencies means your package only includes the libraries it must use to function immediately, keeping the install footprint small. Declaring some dependencies as peerDependencies lets the consumer provide those libraries themselves, so you don't ship duplicates or force a particular version. Marking non-essential pieces as optionalDependencies allows npm to continue installing even if those optional parts can't be resolved, reducing the risk of a failed install. Lazy-loading or dynamically importing heavy modules means you only pull them in when a feature is invoked, not at package load time, further trimming the immediate install cost and improving compatibility with diverse environments. Why other approaches aren't ideal: bundling all dependencies into your package bloats the install size and can cause version conflicts or licensing concerns for users. Requiring heavy dependencies for all setups can break installations in environments that don't have those packages or can't accommodate them. Limiting installation to development environments would prevent end users from using the package at all in production. In practice, you'd structure your package.json to favor minimal runtime needs, expose heavy features behind lazy imports, and use peer and optional dependencies to give the consumer control over what gets installed and loaded.

9. How can generator authors reduce cross-platform template path issues?

A. Use this.templatePath and this.destinationPath with path.join to build paths

B. Hard-code paths to known directories

C. Avoid using path module altogether

D. Use absolute paths for all platforms

Cross-platform path handling comes down to building file paths in a portable way using the generator's path helpers and a path-joining utility. Using this.templatePath and this.destinationPath gives you reliable base locations for templates and where files should be written. When you combine those bases with path.join, you get properly formed paths that work on Windows, macOS, and Linux without worrying about different separators or accidental missing slashes. This approach prevents common pitfalls like incorrect or hard-coded separators, or mislocated template files, and it keeps the generator flexible no matter where it's run from. Hard-coding paths ties the generator to a specific filesystem layout and platform, which breaks portability. Skipping the path module and string-concatenating paths risks incorrect separators and broken paths on some systems. Relying on absolute paths for everything reduces flexibility and can point to locations that don't exist on a user's machine. Using the generator's built-in path helpers with path.join is the robust, portable way to handle templates and destinations.

10. Which elements are used to file records?

A. Subject ID number; Consecutive number; Issuing authority

B. Subject Name; Date of birth; Issuing authority

C. Subject ID number; Date of birth; File number

D. Subject ID number; Consecutive number; Receiving authority

When filing records, you want identifiers that are stable, unique, and traceable to their source. The subject ID number provides a unique identifier for the individual, which helps avoid confusion when names are similar or common. A consecutive number gives each file its own distinct label in a simple, predictable sequence, making organization and retrieval straightforward and preventing any possibility of duplicate files. The issuing authority shows who created or approved the file, establishing provenance and accountability for the record. Using a subject name and date of birth can lead to mismatches because names repeat and dates of birth aren't unique. The receiving authority indicates where the file goes, not where it came from, so it doesn't help establish provenance. A file number alone could be ambiguous across systems, whereas a consecutive number keeps the filing system clean and unambiguous. Altogether, subject ID number, consecutive file number, and issuing authority form a robust filing scheme.

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://yeoman.examzify.com

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