

Test Results and Interview Guide

Candidate: **Elizabeth Wantsajob**
Assessment: Node.JS
Completed: June 28, 2026
Prepared for: Sara Maple
Example Company

What's Included

- Overall Score
- Competency Summary Table
- Comparison Matrix
- Detailed Competency Results with Interview Guide

Important Note: The Node.JS assessment measures key factors related to high performance and tenure in this job. Attribute types measured vary by test, but can include cognitive ability, skills, knowledge, personality characteristics, emotional intelligence, and past behavioral history. This report includes a one page summary, followed by detailed results with an embedded interview guide. Note that these results should always be used as a part of a balanced candidate selection process that includes independent evaluation steps, such as interviews and reference checks.

Overall

Candidate	Score	Interpretation
Elizabeth Wantsajob beth.wantsajob@gmail.com Node.JS June 28, 2026	70	

The candidate demonstrates a solid, working knowledge of Node.JS, including proficiency with asynchronous programming patterns, package management, error handling, and REST API development using common frameworks. They are likely capable of independently handling most entry-level to mid-level development responsibilities, with only occasional guidance needed for complex or specialized tasks.

Key

- Candidate Score
- Higher Risk
- Lower Risk

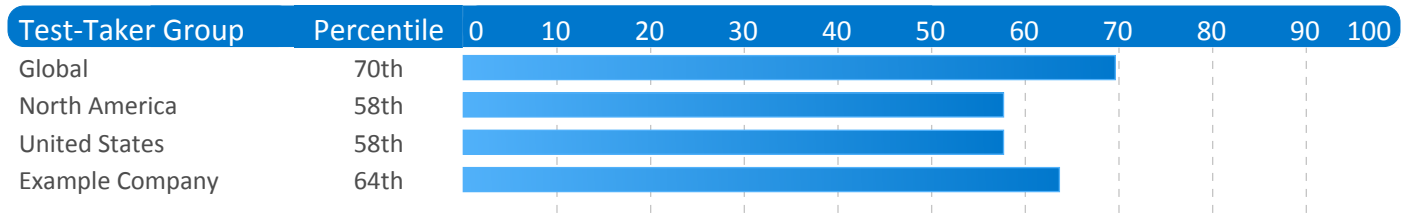
Competency Summary

Competency	Score	Interpretation
Skills/Knowledge (relates to immediate readiness)		
Asynchronous Programming (Callbacks, Promises, and Async/Await)	63	
Asynchronous Programming (Callbacks, Promises, and Async/Await) (Coding Tasks)	62	
Environment Configuration, Process Management, and Debugging (Coding Tasks)	62	
Error Handling (Coding Tasks)	62	
File System and Built-in Core Modules (Coding Tasks)	62	
HTTP Servers, Express.js, and REST API Development (Coding Tasks)	62	
Modules and Package Management (CommonJS, ES Modules, and npm) (Coding Tasks)	62	
Environment Configuration, Process Management, and Debugging	90	
Error Handling	67	
File System and Built-in Core Modules	79	
HTTP Servers, Express.js, and REST API Development	64	
Modules and Package Management (CommonJS, ES Modules, and npm)	96	

↑ Importance to Job

Comparison

Percentile scores indicate how the candidate compares to other test-takers within various groups. The candidate scored equal to or better than the fraction of test-takers indicated by the percentile.



Artificial Intelligence (AI) Generated Scores

This table includes one or more scores derived from a large language model AI query. AI-derived scores are non-deterministic. That is, they are not precisely repeatable. Therefore, these scores should always be treated as supplementary information and should never be used exclusively or compared to hard cutoff values.

Estimated Value	Score	Confidence	Interpretation
Knowledge, Skills, and Abilities Summary	-	-	<p>Summary Points (AI):</p> <ul style="list-style-type: none"> (Generic Text for Sample Report) Strong performer in Drag and Drop Files tasks, indicating comfort with file management and basic computer interactions. Demonstrates solid numerical accuracy in Recognizing and Confirming Numbers, a valuable asset in detail-oriented roles. Moderate overall performance in Analytical Thinking and Attention to Detail, with adequate grammar skills but room for improvement. Struggles with Reading and Analyzing Problems, which may limit effectiveness in roles requiring critical reading and complex problem-solving. Lowest performance in Navigating Between Screens, suggesting difficulty with multi-screen software workflows that could impact productivity in computer-intensive roles. <p>Narrative (AI): Elizabeth Wantsajob demonstrates a mixed profile of knowledge, skills, and abilities across the assessed competencies.</p> <p>Elizabeth shows a strong aptitude in Drag and Drop Files, performing well on this technical task and suggesting she is comfortable with this type of computer interaction. This is a notable strength that would translate well into roles requiring file management and basic computer navigation tasks.</p> <p>In the area of Analytical Thinking and Attention to Detail, Elizabeth performs at a moderate level. She demonstrates solid ability in Recognizing and Confirming Numbers, which suggests she is careful and accurate when working with numerical data — a valuable skill in detail-oriented work environments. Her Grammar performance is adequate but leaves room for improvement, indicating she may occasionally make written communication errors. Her weakest area within this competency is Reading and Analyzing Problems, where she struggled to consistently interpret and work through written problem scenarios. This may impact her effectiveness in roles that require critical reading, written comprehension, or complex problem-solving.</p> <p>Elizabeth's most significant area for development is Navigating Between Screens, where she scored considerably lower than the other competencies. This suggests she may have difficulty efficiently moving through software interfaces or multi-screen workflows, which could slow productivity in roles that rely heavily on navigating computer applications or data entry systems.</p> <p>Overall, Elizabeth brings some useful technical strengths, particularly in file management and numerical accuracy, but would benefit from targeted development in software navigation and analytical problem-solving to be fully effective in roles that demand these skills.</p> <p>Computed on: April 2, 2026, 11:09:49PM EDT</p>

Detail

Candidate: Elizabeth Wantsajob, beth.wantsajob@gmail.com
 Assessment: Node.JS
 Authorized: June 28, 2026, by Sara Maple, Example Company, qamailsaram.mike@hravatar.com
 Started: June 28, 2026, 10:10:13AM EDT
 Completed: June 28, 2026, 10:10:13AM EDT
 Overall Score: 70

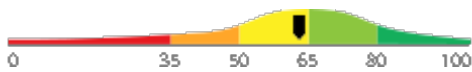
Knowledge and Skills Detail

This section contains a list of job-related knowledge areas and skills that have been evaluated. Low scores in these areas often indicate that additional learning may be required before top performance can be achieved.

Detail
Interview Guide

Asynchronous Programming (Callbacks, Promises, and Async/Await)

Score: 63



Description:

Covers how Node.js handles non-blocking operations using callbacks, Promises, and async/await syntax. Includes understanding of the event loop, how asynchronous code is executed, and how to avoid common pitfalls like callback hell or unhandled promise rejections.

Interpretation:

Candidate appears capable of average job performance in this area with little or no training.

The candidate possesses a partial understanding of Node.JS, demonstrating competence in some core areas such as basic asynchronous patterns, module usage, or HTTP fundamentals, while showing gaps in more advanced topics. They may be able to handle routine development tasks independently but would benefit from guidance on complex features such as database integration, Express.JS, or production deployment.

You have a function that fetches data from a database and another that fetches data from an external API, and neither depends on the other. How would you run both operations at the same time and wait for both to finish before continuing, using `async/await`?



1
Runs operations sequentially; does not use `Promise.all` or equivalent concurrent approach.

2
Identifies the need for concurrency and uses `Promise.all` correctly but omits error handling.

3
Uses `Promise.all` with `async/await` fluently; handles errors and explains the performance benefit clearly.

Can you explain what it means for Node.js to be 'non-blocking,' and give an example of how you would write an asynchronous function using either a callback, a Promise, or `async/await`?



1
Confuses synchronous and asynchronous concepts; cannot provide a working example.

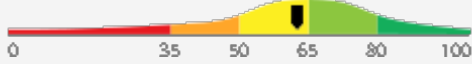
2
Explains non-blocking at a surface level; provides a basic but mostly correct example.

3
Clearly explains non-blocking I/O; gives a correct, well-structured example with error handling.

Detail Interview Guide

Asynchronous Programming (Callbacks, Promises, and Async/Await) (Coding Tasks)

Score: 62



Description:

Covers the use of pointers to reference and manipulate memory addresses, along with dynamic memory allocation and deallocation using malloc, calloc, realloc, and free. Includes pointer arithmetic, dereferencing, and avoiding common issues like memory leaks and dangling pointers.

Interpretation:

The candidate exhibits average writing skills, which can hinder high performance in some jobs.

The candidate possesses a moderate working knowledge of C programming, demonstrating familiarity with core concepts including data types, control flow, functions, and basic file I/O. They may require some guidance when working with more advanced topics such as dynamic memory allocation, modular design, or debugging complex logic.

Overall AI Score:	65.0
Lines of Code:	15.0
Syntax Errors:	5.0
AI Confidence Level:	50
Match with Ideal Response (AI):	30.0
Structure:	50.0
Syntax:	30.0

Please see below to view the essay submitted.

Walk me through how you would dynamically allocate memory for an array of 10 integers, use it, and then properly release it. What issues might arise if you don't follow best practices?



1

Cannot write correct allocation code; unaware of free() or memory leak risks.



2

Writes mostly correct malloc/free code; identifies memory leaks but misses other risks.



3



4

Correct malloc, use, and free; identifies leaks, dangling pointers, and NULL check on allocation.



5

Can you explain what a pointer is in C and describe a situation where you would use one?



1

Vague or incorrect definition; cannot describe a practical use case.



2

Correct basic definition; gives a simple but valid use case with some gaps.



3



4

Clear definition with accurate use case; mentions address storage, dereferencing, or dynamic memory.



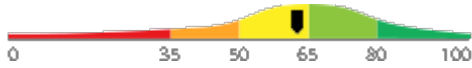
5

Detail

Interview Guide

Environment Configuration, Process Management, and Debugging (Coding Tasks)

Score: 62



Description:

Covers the use of pointers to reference and manipulate memory addresses, along with dynamic memory allocation and deallocation using malloc, calloc, realloc, and free. Includes pointer arithmetic, dereferencing, and avoiding common issues like memory leaks and dangling pointers.

Interpretation:

The candidate exhibits average writing skills, which can hinder high performance in some jobs.

The candidate possesses a moderate working knowledge of C programming, demonstrating familiarity with core concepts including data types, control flow, functions, and basic file I/O. They may require some guidance when working with more advanced topics such as dynamic memory allocation, modular design, or debugging complex logic.

Overall AI Score:	65.0
Lines of Code:	15.0
Syntax Errors:	5.0
AI Confidence Level:	50
Match with Ideal Response (AI):	30.0
Structure:	50.0
Syntax:	30.0

Please see below to view the essay submitted.

Walk me through how you would dynamically allocate memory for an array of 10 integers, use it, and then properly release it. What issues might arise if you don't follow best practices?



1

Cannot write correct allocation code; unaware of free() or memory leak risks.



2

Writes mostly correct malloc/free code; identifies memory leaks but misses other risks.



3



4

Correct malloc, use, and free; identifies leaks, dangling pointers, and NULL check on allocation.



5

Can you explain what a pointer is in C and describe a situation where you would use one?



1

Vague or incorrect definition; cannot describe a practical use case.



2

Correct basic definition; gives a simple but valid use case with some gaps.



3



4

Clear definition with accurate use case; mentions address storage, dereferencing, or dynamic memory.



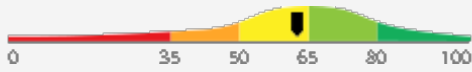
5

Detail

Interview Guide

Error Handling (Coding Tasks)

Score: 62



Description:

Covers the use of pointers to reference and manipulate memory addresses, along with dynamic memory allocation and deallocation using malloc, calloc, realloc, and free. Includes pointer arithmetic, dereferencing, and avoiding common issues like memory leaks and dangling pointers.

Interpretation:

The candidate exhibits average writing skills, which can hinder high performance in some jobs.

The candidate possesses a moderate working knowledge of C programming, demonstrating familiarity with core concepts including data types, control flow, functions, and basic file I/O. They may require some guidance when working with more advanced topics such as dynamic memory allocation, modular design, or debugging complex logic.

Overall AI Score:	65.0
Lines of Code:	15.0
Syntax Errors:	5.0
AI Confidence Level:	50
Match with Ideal Response (AI):	30.0
Structure:	50.0
Syntax:	30.0

Please see below to view the essay submitted.

Walk me through how you would dynamically allocate memory for an array of 10 integers, use it, and then properly release it. What issues might arise if you don't follow best practices?



1

Cannot write correct allocation code; unaware of free() or memory leak risks.



2

Writes mostly correct malloc/free code; identifies memory leaks but misses other risks.



3



4

Correct malloc, use, and free; identifies leaks, dangling pointers, and NULL check on allocation.



5

Can you explain what a pointer is in C and describe a situation where you would use one?



1

Vague or incorrect definition; cannot describe a practical use case.



2

Correct basic definition; gives a simple but valid use case with some gaps.



3



4

Clear definition with accurate use case; mentions address storage, dereferencing, or dynamic memory.

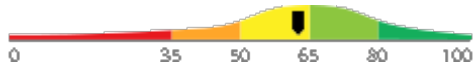


5

Detail Interview Guide

File System and Built-in Core Modules (Coding Tasks)

Score: 62



Description:

Covers the use of pointers to reference and manipulate memory addresses, along with dynamic memory allocation and deallocation using malloc, calloc, realloc, and free. Includes pointer arithmetic, dereferencing, and avoiding common issues like memory leaks and dangling pointers.

Interpretation:

The candidate exhibits average writing skills, which can hinder high performance in some jobs.

The candidate possesses a moderate working knowledge of C programming, demonstrating familiarity with core concepts including data types, control flow, functions, and basic file I/O. They may require some guidance when working with more advanced topics such as dynamic memory allocation, modular design, or debugging complex logic.

Overall AI Score:	65.0
Lines of Code:	15.0
Syntax Errors:	5.0
AI Confidence Level:	50
Match with Ideal Response (AI):	30.0
Structure:	50.0
Syntax:	30.0

Please see below to view the essay submitted.

Walk me through how you would dynamically allocate memory for an array of 10 integers, use it, and then properly release it. What issues might arise if you don't follow best practices?



1

Cannot write correct allocation code; unaware of free() or memory leak risks.



2

Writes mostly correct malloc/free code; identifies memory leaks but misses other risks.



3



4

Correct malloc, use, and free; identifies leaks, dangling pointers, and NULL check on allocation.



5

Can you explain what a pointer is in C and describe a situation where you would use one?



1

Vague or incorrect definition; cannot describe a practical use case.



2

Correct basic definition; gives a simple but valid use case with some gaps.



3



4

Clear definition with accurate use case; mentions address storage, dereferencing, or dynamic memory.

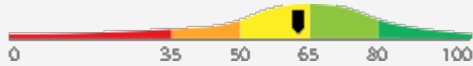


5

Detail Interview Guide

HTTP Servers, Express.js, and REST API Development (Coding Tasks)

Score: 62



Description:

Covers the use of pointers to reference and manipulate memory addresses, along with dynamic memory allocation and deallocation using malloc, calloc, realloc, and free. Includes pointer arithmetic, dereferencing, and avoiding common issues like memory leaks and dangling pointers.

Interpretation:

The candidate exhibits average writing skills, which can hinder high performance in some jobs.

The candidate possesses a moderate working knowledge of C programming, demonstrating familiarity with core concepts including data types, control flow, functions, and basic file I/O. They may require some guidance when working with more advanced topics such as dynamic memory allocation, modular design, or debugging complex logic.

Overall AI Score:	65.0
Lines of Code:	15.0
Syntax Errors:	5.0
AI Confidence Level:	50
Match with Ideal Response (AI):	30.0
Structure:	50.0
Syntax:	30.0

Please see below to view the essay submitted.

Walk me through how you would dynamically allocate memory for an array of 10 integers, use it, and then properly release it. What issues might arise if you don't follow best practices?



1

Cannot write correct allocation code; unaware of free() or memory leak risks.



2

Writes mostly correct malloc/free code; identifies memory leaks but misses other risks.



3



4

Correct malloc, use, and free; identifies leaks, dangling pointers, and NULL check on allocation.



5

Can you explain what a pointer is in C and describe a situation where you would use one?



1

Vague or incorrect definition; cannot describe a practical use case.



2

Correct basic definition; gives a simple but valid use case with some gaps.



3



4

Clear definition with accurate use case; mentions address storage, dereferencing, or dynamic memory.

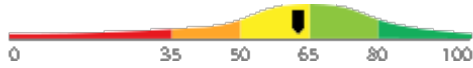


5

Detail Interview Guide

Modules and Package Management (CommonJS, ES Modules, and npm) (Coding Tasks)

Score: 62



Description:

Covers the use of pointers to reference and manipulate memory addresses, along with dynamic memory allocation and deallocation using malloc, calloc, realloc, and free. Includes pointer arithmetic, dereferencing, and avoiding common issues like memory leaks and dangling pointers.

Interpretation:

The candidate exhibits average writing skills, which can hinder high performance in some jobs.

The candidate possesses a moderate working knowledge of C programming, demonstrating familiarity with core concepts including data types, control flow, functions, and basic file I/O. They may require some guidance when working with more advanced topics such as dynamic memory allocation, modular design, or debugging complex logic.

Overall AI Score:	65.0
Lines of Code:	15.0
Syntax Errors:	5.0
AI Confidence Level:	50
Match with Ideal Response (AI):	30.0
Structure:	50.0
Syntax:	30.0

Please see below to view the essay submitted.

Walk me through how you would dynamically allocate memory for an array of 10 integers, use it, and then properly release it. What issues might arise if you don't follow best practices?



1

Cannot write correct allocation code; unaware of free() or memory leak risks.



2

Writes mostly correct malloc/free code; identifies memory leaks but misses other risks.



3



4

Correct malloc, use, and free; identifies leaks, dangling pointers, and NULL check on allocation.



5

Can you explain what a pointer is in C and describe a situation where you would use one?



1

Vague or incorrect definition; cannot describe a practical use case.



2

Correct basic definition; gives a simple but valid use case with some gaps.



3



4

Clear definition with accurate use case; mentions address storage, dereferencing, or dynamic memory.



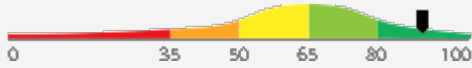
5

Detail

Interview Guide

Environment Configuration, Process Management, and Debugging

Score: 90



Description:

Covers how to manage application settings using environment variables and the process object, how to use tools like dotenv to load configuration, and how to run and monitor Node.js applications in production using process managers like PM2. Includes basic debugging techniques using Node's built-in debugger or console-based strategies.

Interpretation:

Candidate should achieve superior job performance in this area with little or no training.

The candidate exhibits an advanced and comprehensive mastery of Node.js environment configuration, process management, and debugging. They are well-equipped to manage application settings, oversee production deployments using process managers, and employ sophisticated debugging techniques with a high degree of confidence and accuracy.

If your Node.js application crashes in production, how would you set it up so that it automatically restarts, and how would you investigate what caused the crash?



1

Has no strategy for auto-restart or crash investigation beyond basic console logging.



2

Mentions a process manager like PM2 for auto-restart and log files or console output for investigation.



3



4

Describes PM2 or equivalent, structured logging, reviewing error logs, and may mention monitoring or alerting tools.



5

How would you store a sensitive value like a database password in a Node.js application so it is not hardcoded in your source code?



1

Suggests hardcoding credentials or is unaware of environment variables.



2

Describes using environment variables and process.env correctly; may mention .env files.



3



4

Explains using a .env file with dotenv, accessing values via process.env, and keeping .env out of version control.



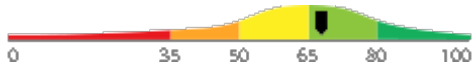
5

Detail

Interview Guide

Error Handling

Score: 67



Description:

Covers strategies for catching and managing errors in Node.js applications, including try/catch blocks for synchronous and async code, handling error events on streams or event emitters, and managing unhandled promise rejections. Includes writing code that fails gracefully and provides useful error information.

Interpretation:

Candidate should achieve above average job performance in this area with little or no training.

The candidate demonstrates a solid and competent understanding of error handling in Node.js, including the use of try/catch for both synchronous and asynchronous code, handling error events, and managing unhandled promise rejections. They are generally capable of writing resilient code that fails gracefully and surfaces useful error information.

How would you handle errors in an Express.js application so that any unexpected error in a route handler is caught and returns a consistent JSON error response to the client instead of crashing the server?



1

Unaware of Express error-handling middleware or how to pass errors to it using next(err).



2

Describes a centralized error-handling middleware with four parameters but may miss async error forwarding.



3



4

Correctly implements error-handling middleware, wraps async routes to forward errors, and returns structured JSON responses with appropriate status codes.



5

If an error occurs inside an async function, how do you make sure your application catches it instead of crashing or silently failing?



1

Cannot describe try/catch in async functions or any strategy for catching async errors.



2

Describes using try/catch inside an async function correctly but may miss promise rejection edge cases.



3



4

Explains try/catch in async/await, .catch() on Promises, and may mention process-level handlers for uncaught rejections.



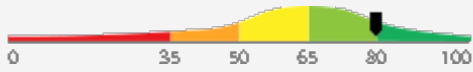
5

Detail

Interview Guide

File System and Built-in Core Modules

Score: 79



Description:

Covers using Node.js built-in modules to perform common tasks such as reading and writing files with the fs module, working with file paths using the path module, and handling streams and buffers for efficient data processing. Includes both synchronous and asynchronous file operations.

Interpretation:

Candidate should achieve above average job performance in this area with little or no training.

The candidate exhibits a solid and well-rounded knowledge of Node.js file system and built-in core modules, including proficient use of the fs and path modules and an understanding of both synchronous and asynchronous patterns. They are likely capable of handling most file operation tasks independently with only occasional need for reference.

You need to build a Node.js script that reads a large log file and processes it line by line without loading the entire file into memory. How would you approach this?



1

Attempts to read the entire file at once; unaware of streams or readline module.



2

Identifies streams or the readline module as the right approach and describes a workable solution.



3



4

Clearly explains using a readable stream with the readline interface or pipe, explains why it is memory-efficient, and handles errors.



5

How would you read the contents of a text file in Node.js and print them to the console? What built-in module would you use?



1

Cannot name the fs module or describe how to read a file.



2

Correctly identifies fs and uses readfile or readfileSync with mostly correct syntax.



3



4

Provides a correct async example with error handling; may contrast async vs. sync and mention encoding options.



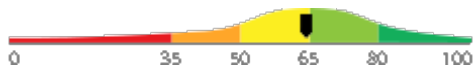
5

Detail

Interview Guide

HTTP Servers, Express.js, and REST API Development

Score: 64


Description:

Covers creating HTTP servers using Node's built-in http module and building REST APIs using Express.js. Includes defining routes, handling requests and responses, using middleware for tasks like parsing request bodies or validating input, and returning appropriate HTTP status codes.

Interpretation:

Candidate appears capable of average job performance in this area with little or no training.

The candidate has a moderate understanding of HTTP server creation and REST API development using Express.js, demonstrating familiarity with core concepts such as routing, request and response handling, and basic middleware. However, gaps in knowledge may exist in more advanced or nuanced areas of the subject.

In an Express.js application, how would you create a REST API endpoint that accepts a JSON body in a POST request, validates that a required field is present, and returns an appropriate error response if it is missing?



1

Cannot set up a POST route or parse a JSON body; unaware of how to send error status codes.



2

Sets up the route and parses the body correctly but handles validation or error responses incompletely.



3



4



5

Correctly configures express.json() middleware, validates input, and returns meaningful status codes and error messages.

How would you create a simple web server in Node.js that responds with the text 'Hello World' when someone visits the homepage?



1

Cannot describe how to create a server; unaware of http module or Express basics.



2

Provides a working example using either the http module or Express but may have minor syntax errors.



3



4



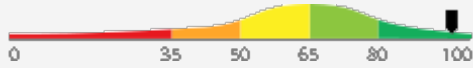
5

Gives a clean, correct example; may contrast the http module with Express and explain when to use each.

Detail Interview Guide

Modules and Package Management (CommonJS, ES Modules, and npm)

Score: 96



Description:

Covers how Node.js organizes code into reusable modules using `require/module.exports` (CommonJS) or `import/export` (ES Modules). Includes using npm to install and manage third-party packages, understanding `package.json`, managing dependencies, and using npm scripts to automate tasks.

Interpretation:

Candidate should achieve superior job performance in this area with little or no training.

The candidate demonstrates a comprehensive and advanced understanding of Node.js module systems and package management. They are highly proficient with CommonJS and ES Modules, `package.json` configuration, dependency management strategies, and npm script automation, and are well-equipped to apply these skills effectively in professional Node.js development.

What is the difference between dependencies and devDependencies in `package.json`, and how would you use an npm script to automate running your application or tests?



1

Cannot distinguish dependencies from devDependencies; unfamiliar with npm scripts.



2

Correctly distinguishes dependency types and can write a basic npm script entry.



3



4

Clearly explains both concepts, gives practical examples of npm scripts, and may mention environment-specific installs like `npm ci`.



5

How do you share code between two files in a Node.js project, and how would you add a third-party package to your project so you can use it in your code?



1

Cannot describe module exports or npm install; shows little awareness of how Node.js projects are structured.



2

Correctly describes `require/module.exports` or `import/export` and basic npm install usage.



3



4

Explains both CommonJS and ES Module syntax, describes `package.json` roles, and mentions devDependencies vs. dependencies.



5

IT Coding Tasks

During the assessment, the candidate was asked to write one or more programs or scripts. Their responses are included below for review.

Question or Task	Response
<p>Complete the provided partial C program by filling in the missing sections marked with TODO comments. Your completion must use standard C keywords and libraries.</p> <p>The program declares a function called <code>duplicate_array</code> that:</p> <ol style="list-style-type: none"> 1. Takes a const int pointer to a source array and its length as parameters. 2. Uses <code>calloc</code> to allocate a new int array of the same length. 3. Returns NULL if <code>calloc</code> fails. 4. Copies each element from the source array into the new array using pointer arithmetic (not array subscript notation). 5. Returns the pointer to the newly allocated copy. <p>In main, the program:</p> <ol style="list-style-type: none"> 1. Declares and initializes a stack array of 4 integers with values 5, 15, 25, 35. 2. Calls <code>duplicate_array</code> to create a heap-allocated copy. 3. Checks for NULL and prints an error and returns 1 if the call failed. 4. Prints each element of the duplicate using a loop. 5. Frees the duplicate array. <p>Use appropriate indentation, common C coding conventions, and add brief inline comments where needed. Type your completed source code as your response.</p>	<pre>#include <stdio.h> #include <stdlib.h> int *duplicate_array(const int *src, int length) { /* TODO: Use calloc to allocate a new array of 'length' integers, return NULL if calloc fails, copy elements from src using pointer arithmetic, and return the new pointer. */ calloc(303); } int main(void) { /* TODO: Declare and initialize a stack array of 4 integers: 5, 15, 25, 35, then call duplicate_array and store the result. Check for NULL and print an error message returning 1 if it failed. */ array[4]={5,15,25,35}; int i; /* Print each element of the duplicate */ for (i = 0; i < 4; i++) { printf("duplicate[%d] = %d\n", i, *(duplicate + i)); } /* Free the duplicate array */ free(duplicate); return 0; }</pre>

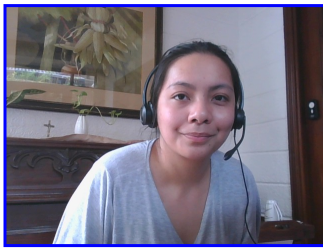
Comments (AI): The code segment has several syntax errors and incomplete implementation. The `duplicate_array` function does not correctly allocate memory or copy elements. The main function has syntax errors and does not properly call the `duplicate_array` function. However, the structure and intent of the code are somewhat clear, and the code attempts to follow the requirements.

Identity Confirmation Photos

The following photos of the candidate and any identification were uploaded during the assessment session.

Photo Analysis Results

- Risk:	Medium risk of cheating based on image inconsistencies
- Percent match among processed faces	100%
- Total images processed	17
- Total images with valid faces	14 (82%)
- Total pairs of faces compared	13
- Pairs in which faces matched	13 (100%)



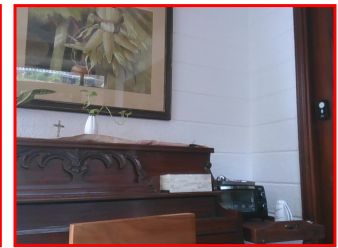
Pre/Post-Test Photo



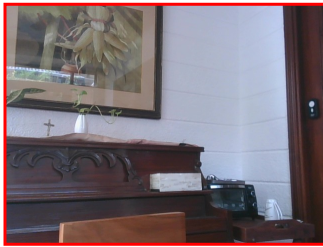
ID Photo



In-Test Error Detected (No Face Detected)



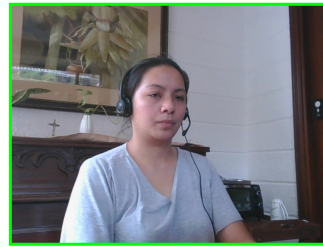
In-Test Error Detected (No Face Detected)



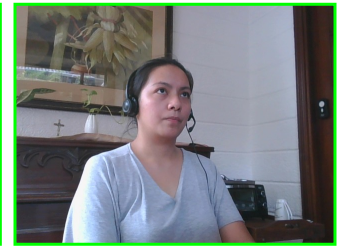
In-Test Error Detected (No Face Detected)



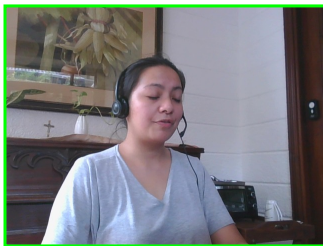
In-Test Photo



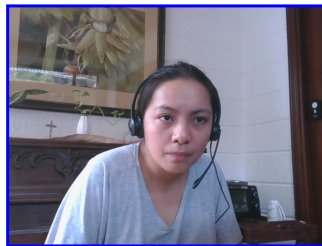
In-Test Photo



In-Test Photo



In-Test Photo



Pre/Post-Test Photo

Resume or CV

[Summary](#)[Updated on](#)

Motivated career professional with extensive experience in office administration and management. Proven track record of improving efficiency, reducing costs, and enhancing office operations through strategic initiatives and technology implementation.

Objective

I am seeking a role where I can use my many skills and my exceptional judgment and empathy for customers to make a difference to a growing company.

Education

- Associate of Applied Science in Office Administration, Portland Community College, 2020

Experience

- General Office Clerk, Paramount Office Management, 09/2023 – Present
- Administrative Assistant, Global Enterprises Inc., 04/2021 – 08/2023
- Administrative Assistant, Innovative Business Solutions Ltd., 07/2019 – 03/2021

Other Qualifications

- Microsoft Office Specialist (MOS) Certification
- Certified Administrative Professional (CAP)
- International Association of Administrative Professionals (IAAP) Certification

Report Preparation Notes

- Hiring decisions should never be based on a single source of information. The most effective use of this assessment report is as a part of a multi-faceted program of candidate evaluation that includes resume review, interviews, and reference checks.
- Overall vs Percentiles Scores: The overall score reflects the success in the test, based on the mean (average) and standard deviation of the test scores. The percentile score reflects the percentage of test-takers who scored equal or below this overall score. We recommend you use the Overall Score as your primary evaluation criteria. However, percentile scores can often be useful in comparing specific candidates against one another and with a group, such as for test takers in a certain organization or within a certain account.
- Note that comparison information is calculated based on completed instances of this assessment at that time the assessment is scored. As additional instances are completed, the comparative data may change. You can always update a report to the current values by clicking on 'Recalculate Percentiles' within the online results viewing pages at www.hravatar.com.
- Most competency scores are norm-based, which means that they can be interpreted in terms of their distance from the average or mean score. For all scales, a score equal to the mean receives a score of 65 and scores above and below this value are set so that a score change of 15 equals one standard deviation.
- For linear competencies, higher is better across the entire scale. For these scales a score between 65 and 80 (light green) represents 0 to 1 standard deviation above the mean and a score above 80 (dark green) represents more than one standard deviation above the mean. Similarly, a score of 50 - 65 (yellow) represents 0 to 1 standard deviation below the mean, while a score of 35 - 50 (orange) equates to 1 to 2 standard deviations below the mean, and a score below 35 represents more than 2 standard deviations below the mean.
- Sim ID: 20809-1, Key: 0-0, Rpt: 68, Prd: 9631, Created: 2026-06-28 10:10 EDT
- UA: Mozilla/5.0 (Windows NT 6.3; Trident/7.0; Touch; rv:11.0) like Gecko

Score Calculation Detail

The following table provides a summary of how the overall score was calculated from each of the individual competency scores. First, all competency scores are calculated on a scale of 0-100. Note that some competencies use their color category rather than their actual numeric score in the overall calculation. For these, a standard score associated with the assigned color category is used in the overall score calculation rather than the actual numeric score. This is reflected in the "Score Value Used" column. Next, a weighted average of scores is computed using individual competency weights, typically set using job analysis data provided by the US Government Occupational Information Network (O*Net).

Competency	Score	How applied to overall	Score Value Used	Weight (%)
Asynchronous Programming (Callbacks, Promises, and Async/Await)	63.3998	Numeric Score	63.3998	8.3333
Asynchronous Programming (Callbacks, Promises, and Async/Await) (Coding Tasks)	62.9784	Numeric Score	62.9784	8.3333
Environment Configuration, Process Management, and Debugging	90.3417	Numeric Score	90.3417	8.3333
Environment Configuration, Process Management, and Debugging (Coding Tasks)	62.9784	Numeric Score	62.9784	8.3333
Error Handling	67.9894	Numeric Score	67.9894	8.3333
Error Handling (Coding Tasks)	62.9784	Numeric Score	62.9784	8.3333
File System and Built-in Core Modules	79.9084	Numeric Score	79.9084	8.3333
File System and Built-in Core Modules (Coding Tasks)	62.9784	Numeric Score	62.9784	8.3333
HTTP Servers, Express.js, and REST API Development	64.7650	Numeric Score	64.7650	8.3333
HTTP Servers, Express.js, and REST API Development (Coding Tasks)	62.9784	Numeric Score	62.9784	8.3333
Modules and Package Management (CommonJS, ES Modules, and npm)	96.3878	Numeric Score	96.3878	8.3333
Modules and Package Management (CommonJS, ES Modules, and npm) (Coding Tasks)	62.9784	Numeric Score	62.9784	8.3333
Weighted Average:				70.0552
Final Overall Score:				70

Notes

(This area is intentionally blank - it's reserved as space for your notes.)