hr | avatar
Illuminating Talent

Test Results and Interview Guide

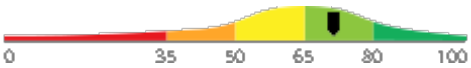
Candidate: **Elizabeth Wantsajob**
Assessment: Unix Bash Scripting
Completed: June 30, 2026
Prepared for: Sara Maple
Example Company

What's Included

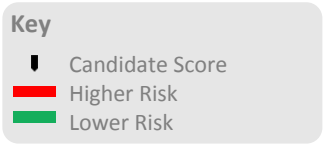
- Overall Score
- Competency Summary Table
- Comparison Matrix
- Detailed Competency Results with Interview Guide

Important Note: The Unix Bash Scripting 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.

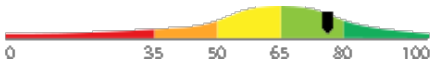
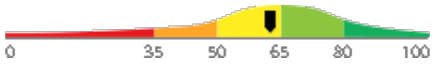
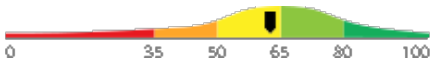
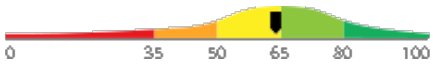
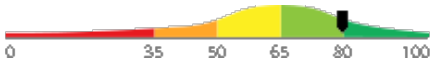
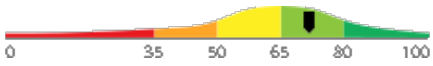
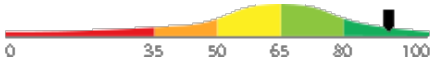
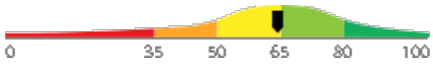
Overall

Candidate	Score	Interpretation
Elizabeth Wantsajob beth.wantsajob@gmail.com Unix Bash Scripting June 30, 2026	71	

The candidate exhibits a solid and competent understanding of Unix Bash scripting, including proficiency with control structures, functions, text-processing tools, environment variables, error handling, and file system operations. They are well-equipped to independently write, debug, and maintain a broad range of scripts, with only occasional gaps in more advanced or specialized areas.



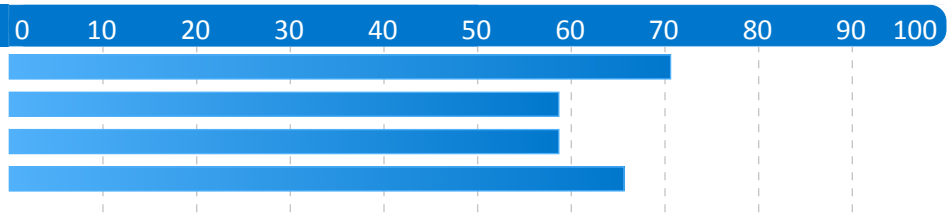
Competency Summary

Competency	Score	Interpretation
Skills/Knowledge (relates to immediate readiness)		
Control Structures: Conditionals and Loops	76	
Control Structures: Conditionals and Loops (Coding Tasks)	62	
Variables, Data Types, and Parameters (Coding Tasks)	62	
Error Handling and Exit Codes	64	
File Operations and Permissions	79	
Input, Output, Redirection, and Piping	72	
Text Processing with grep, sed, and awk	90	
Variables, Data Types, and Parameters	64	

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.

Test-Taker Group	Percentile
Global	71st
North America	59th
United States	59th
Example Company	66th



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	-	-	Summary Points (AI): • (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. Narrative (AI): Elizabeth Wantsajob demonstrates a mixed profile of knowledge, skills, and abilities across the assessed competencies. 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. 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. 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. 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. Computed on: April 2, 2026, 11:09:49PM EDT

Detail

Candidate:	Elizabeth Wantsajob, beth.wantsajob@gmail.com
Assessment:	Unix Bash Scripting
Authorized:	June 30, 2026, by Sara Maple, Example Company, qamailsaram.mike@hravatar.com
Started:	June 30, 2026, 5:04:42PM EDT
Completed:	June 30, 2026, 5:04:42PM EDT
Overall Score:	71

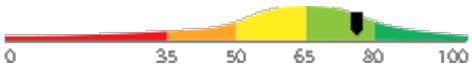
Knowledge and Skills Detail

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

Control Structures: Conditionals and Loops

Score: 76



Description:

Covers the use of if/else statements, case statements, and loops (for, while, until) to control the flow of a Bash script. Includes correct syntax for conditions and comparison operators for strings and numbers.

Interpretation:

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

The candidate demonstrates a solid working knowledge of Bash control structures, including conditional statements, case statements, and loop constructs. They are generally proficient with conditional syntax and comparison operators, though some gaps in advanced usage or edge cases may exist.

Describe a situation where you would choose a while loop over a for loop in a Bash script, and write a brief example of each.

1	2	3	4	5
Cannot clearly distinguish between the two or produces non-functional examples.		Explains the difference and writes one correct example but struggles with the other.		Clearly contrasts both loops with accurate, practical examples and explains appropriate use cases for each.

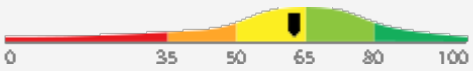
Can you walk me through how you would write an if statement in Bash to check whether a file exists?

1	2	3	4	5
Cannot recall if statement syntax or file test operators; example is non-functional.		Produces a working if statement but uses incorrect or inconsistent bracket syntax.		Writes correct if statement using -f or -e flag, proper brackets, and explains the logic clearly.

Detail

Control Structures: Conditionals and Loops (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.

Interview Guide

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	2	3	4	5
Cannot write correct allocation code; unaware of free() or memory leak risks.		Writes mostly correct malloc/free code; identifies memory leaks but misses other risks.		Correct malloc, use, and free; identifies leaks, dangling pointers, and NULL check on allocation.

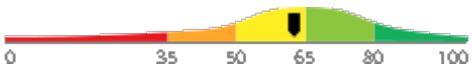
Can you explain what a pointer is in C and describe a situation where you would use one?

1	2	3	4	5
Vague or incorrect definition; cannot describe a practical use case.		Correct basic definition; gives a simple but valid use case with some gaps.		Clear definition with accurate use case; mentions address storage, dereferencing, or dynamic memory.

Detail

Variables, Data Types, and Parameters (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.

Interview Guide

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	2	3	4	5
Cannot write correct allocation code; unaware of free() or memory leak risks.		Writes mostly correct malloc/free code; identifies memory leaks but misses other risks.		Correct malloc, use, and free; identifies leaks, dangling pointers, and NULL check on allocation.

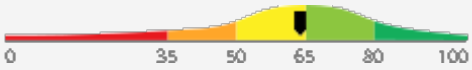
Can you explain what a pointer is in C and describe a situation where you would use one?

1	2	3	4	5
Vague or incorrect definition; cannot describe a practical use case.		Correct basic definition; gives a simple but valid use case with some gaps.		Clear definition with accurate use case; mentions address storage, dereferencing, or dynamic memory.

Detail

Error Handling and Exit Codes

Score: 64



Description:

Covers how Bash scripts report success or failure using exit codes, how to check the result of commands, and how to write scripts that handle errors gracefully. Includes use of exit, $?, set -e, and basic error messaging.

Interpretation:

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

The candidate has a foundational understanding of error handling and exit codes in Unix Bash scripting. They are likely familiar with core concepts such as checking command outcomes and using basic error messaging, but may struggle with more nuanced or complex error handling scenarios.

Interview Guide

How would you write a Bash script that stops running and prints a helpful error message if any command in the script fails?

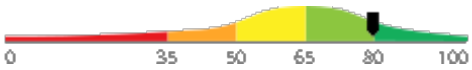
1	2	3	4	5
Cannot describe a method to stop script execution on failure or produce a working example.		Mentions set -e or manual $? checks but cannot combine them into a complete, working error-handling approach.		Uses set -e and/or checks $? after critical commands, includes a descriptive error message, and explains the tradeoffs.

What is an exit code in Bash, and how would you check whether the last command in your script ran successfully?

1	2	3	4	5
Cannot explain exit codes or does not know how to use $? to check command results.		Correctly explains exit codes and $? but cannot describe how to act on the result in a script.		Explains exit codes clearly, uses $? in an if statement, and mentions that 0 means success and non-zero means failure.

File Operations and Permissions

Score: 79



Description:

Covers how to create, read, move, copy, and delete files and directories within a Bash script. Includes checking file properties using test operators and understanding how Unix file permissions affect script behavior.

Interpretation:

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

The candidate shows a solid and competent understanding of Unix Bash file operations and permissions. They are generally proficient in managing files and directories within scripts, applying file test operators, and accounting for the effects of Unix file permissions on script behavior, with only minor gaps in knowledge.

If a Bash script you wrote fails because of a permissions error, what steps would you take to diagnose and fix the problem?

1	2	3	4	5
Cannot describe how to check or change permissions; no mention of chmod or ls -l.		Mentions chmod or ls -l but provides an incomplete or partially incorrect troubleshooting process.		Describes using ls -l to inspect permissions and chmod to fix them, with clear reasoning and correct syntax.

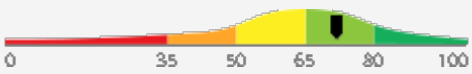
How would you check inside a Bash script whether a file exists and whether it is readable before trying to open it?

1	2	3	4	5
Cannot recall file test operators or produces a non-functional check.		Uses -e or -f correctly but is unsure how to also test for read permission.		Uses both -f and -r test operators correctly within an if statement and explains each flag's purpose.

Detail

Input, Output, Redirection, and Piping

Score: 72



Description:

Covers how to read user input, display output, and redirect standard input, output, and error streams. Includes using pipes to chain commands together and redirect data between processes.

Interpretation:

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

The candidate demonstrates a solid working knowledge of reading input, displaying output, redirecting standard input, output, and error streams, and using pipes to chain commands in Unix Bash scripting. They are expected to handle most practical scenarios competently, with only occasional difficulty in advanced use cases.

Interview Guide

How would you redirect both standard output and standard error from a command to the same log file in a Bash script?

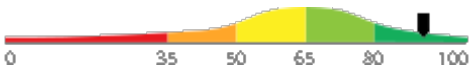
1	2	3	4	5
Cannot recall the syntax for redirecting stderr or conflates stdout and stderr.		Knows 2>&1 or similar syntax but cannot fully construct the correct combined redirection.		Correctly writes command > file.log 2>&1 or equivalent and explains what each part does.

What does the > operator do in a Bash script, and how is it different from >>?

1	2	3	4	5
Cannot distinguish between the two operators or describes them incorrectly.		Correctly explains one operator but is vague or incorrect about the other.		Accurately explains both operators, noting that > overwrites and >> appends, with a practical example.

Text Processing with grep, sed, and awk

Score: 90



Description:

Covers using common Unix text-processing tools within Bash scripts to search, filter, and transform text. Focuses on practical, everyday use of grep for searching, sed for find-and-replace, and awk for field-based text processing.

Interpretation:

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

The candidate exhibits an advanced and comprehensive mastery of Unix text-processing tools within Bash scripting. They are highly proficient in leveraging grep, sed, and awk for sophisticated search, find-and-replace, and field-based text transformation tasks, and are well-suited to serve as a subject matter resource for others.

Describe how you would use sed or awk in a script to extract or replace specific text from a file. Walk me through a practical example.

1	2	3	4	5
Cannot produce a working sed or awk command or confuses their purposes.		Produces a working example for one tool but struggles to explain the syntax or use the other tool.		Demonstrates correct use of both tools with clear examples, explaining syntax like s/old/new/g in sed or field separators in awk.

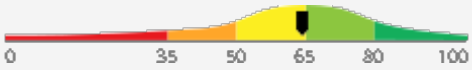
How would you use grep in a Bash script to search for a specific word in a file and print only the lines that contain it?

1	2	3	4	5
Cannot recall grep syntax or constructs a non-functional command.		Writes a mostly correct grep command but omits useful flags or makes minor syntax errors.		Writes a correct grep command, mentions useful flags like -i or -n, and integrates it naturally into a script context.

Detail

Variables, Data Types, and Parameters

Score: 64



Description:

Covers how to declare, assign, and use variables in Bash scripts, including special variables, environment variables, and passing arguments to scripts. Includes string and numeric data handling as well as basic array usage.

Interpretation:

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

The candidate possesses a moderate, foundational understanding of Unix Bash scripting, demonstrating competence in core areas such as basic syntax, variables, and common control structures. However, proficiency in more advanced topics such as process management, regular expressions, scheduling with cron, and complex text-processing tools may be inconsistent or incomplete. With targeted development and practical experience, this candidate has the potential to grow into a capable Bash scripter.

Interview Guide

How would you access and validate arguments passed to a Bash script, and what special variables would you use to do that?

1	2	3	4	5
Cannot name relevant special variables or describe how to access script arguments.		Identifies $1, $2, or $# but provides limited or partially correct validation approach.		Accurately uses $@, $#, and $1–$N with a clear validation strategy such as checking argument count.

Can you explain what a variable is in Bash and show me how you would declare one and use it in a simple script?

1	2	3	4	5
Cannot define a variable or confuses syntax; no working example provided.		Correctly defines and uses a variable but struggles with special variables or arguments.		Clearly explains variables, demonstrates declaration, use, and references special variables like $1 or $@.

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.

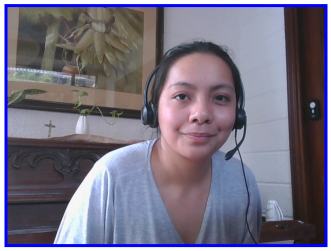
Question or Task	Response

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.
 	
The program declares a function called duplicate_array that:
1. Takes a const int pointer to a source array and its length as parameters.
2. Uses calloc to allocate a new int array of the same length.
3. Returns NULL if calloc 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.
 	
In main, the program:
1. Declares and initializes a stack array of 4 integers with values 5, 15, 25, 35.
2. Calls duplicate_array 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.
 	
Use appropriate indentation, common C coding conventions, and add brief inline comments where needed. Type your completed source code as your response.
	#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;
	}
	
	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

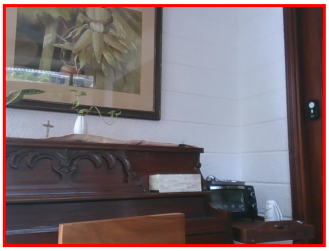
The following photos of the candidate and any identification were uploaded during the assessment session.

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%)

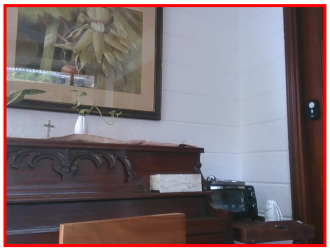


Pre/Post-Test Photo

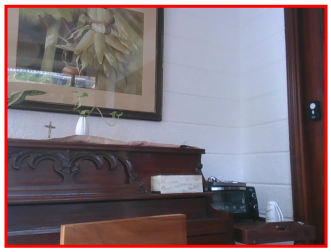
ID Photo



In-Test Error Detected (No Face Detected)

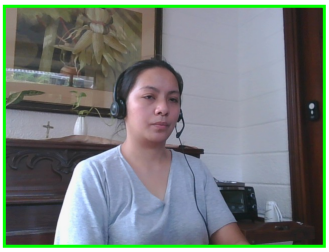


In-Test Error Detected (No Face Detected)

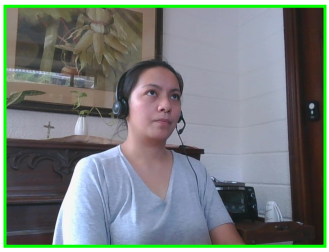


In-Test Error Detected (No Face Detected)

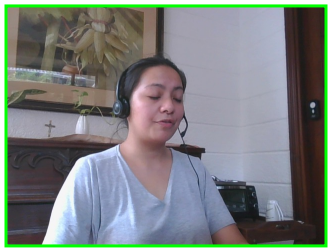
In-Test Photo



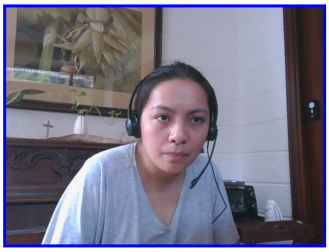
In-Test Photo



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: 20830-1, Key: 0-0, Rpt: 68, Prd: 9653, Created: 2026-06-30 17:04 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 (%)
Control Structures: Conditionals and Loops	76.6005	Numeric Score	76.6005	12.5000
Control Structures: Conditionals and Loops (Coding Tasks)	62.9784	Numeric Score	62.9784	12.5000
Error Handling and Exit Codes	64.3923	Numeric Score	64.3923	12.5000
File Operations and Permissions	79.8960	Numeric Score	79.8960	12.5000
Input, Output, Redirection, and Piping	72.0434	Numeric Score	72.0434	12.5000
Text Processing with grep, sed, and awk	90.8416	Numeric Score	90.8416	12.5000
Variables, Data Types, and Parameters	64.7362	Numeric Score	64.7362	12.5000
Variables, Data Types, and Parameters (Coding Tasks)	62.9784	Numeric Score	62.9784	12.5000
Weighted Average:				71.8084
Final Overall Score:				71

Notes

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