hr | avatar
Illuminating Talent

Test Results and Interview Guide

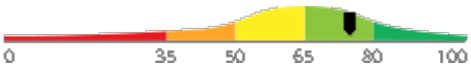
Candidate: **Elizabeth Wantsajob**
Assessment: C# Programming
Completed: June 25, 2026
Prepared for: Sara Maple
Example Company

What's Included

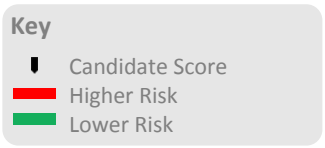
- Overall Score
- Competency Summary Table
- Comparison Matrix
- Detailed Competency Results with Interview Guide

Important Note: The C# Programming 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.

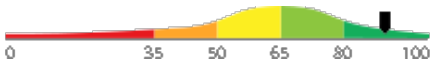
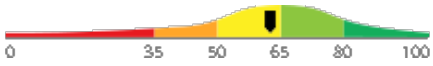
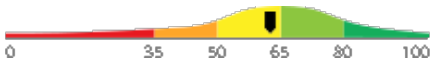
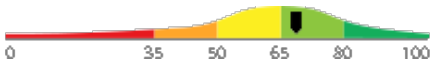
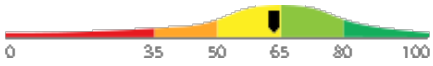
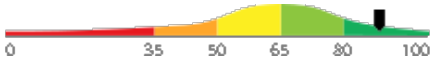
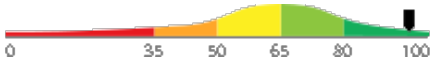
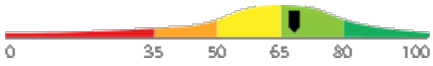
Overall

Candidate	Score	Interpretation
Elizabeth Wantsajob beth.wantsajob@gmail.com C# Programming June 25, 2026	75	

The candidate demonstrates a solid and well-rounded understanding of C# programming, including object-oriented design, exception handling, collections, and data manipulation using LINQ and lambda expressions. Competence in areas such as database connectivity, unit testing, and version control suggests readiness for mid-level development responsibilities with minimal supervision. Minor gaps in advanced topics such as asynchronous programming or dependency injection may exist but are expected to be bridged with practical experience.



Competency Summary

Competency	Score	Interpretation
Skills/Knowledge (relates to immediate readiness)		
Asynchronous Programming with Async and Await	90	
C# Syntax, Data Types, and Control Flow (Coding Tasks)	62	
Object-Oriented Programming (OOP) with Classes and Interfaces (Coding Tasks)	62	
C# Syntax, Data Types, and Control Flow	69	
Collections, LINQ, and Data Manipulation	63	
Database Access with Entity Framework and ADO.NET	88	
Exception Handling and Debugging	95	
Object-Oriented Programming (OOP) with Classes and Interfaces	68	

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	75th
North America	62nd
United States	62nd
Example Company	69th

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:	C# Programming
Authorized:	June 25, 2026, by Sara Maple, Example Company, qamailsaram.mike@hravatar.com
Started:	June 25, 2026, 4:20:31PM EDT
Completed:	June 25, 2026, 4:20:31PM EDT
Overall Score:	75

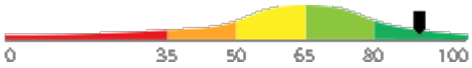
Knowledge and Skills Detail

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 with Async and Await

Score: 90



Description:

Covers how to write non-blocking code in C# using the async and await keywords, Task-based patterns, and how asynchronous programming improves the responsiveness and scalability of applications. This is a regularly required skill in modern business applications that interact with databases, APIs, or file systems.

Interpretation:

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

The candidate exhibits an advanced and comprehensive mastery of asynchronous programming in C#, including async, await, and Task-based patterns. They are highly capable of designing and implementing efficient, non-blocking code that enhances the responsiveness and scalability of complex, modern business applications.

What are some common mistakes developers make when using async and await in C#, and how do you avoid them in your own code?

1	2	3	4	5
Cannot identify any common async mistakes or gives vague, inaccurate answers.		Identifies at least one common mistake such as not awaiting a task or blocking with .Result, with a basic explanation.		Identifies multiple common pitfalls such as async void, deadlocks from .Result/.Wait(), or fire-and-forget issues, with clear explanations and mitigations.

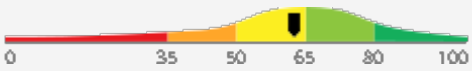
Have you written any asynchronous code in C#? Can you explain in simple terms what async and await do and why you might use them instead of writing regular synchronous code?

1	2	3	4	5
Cannot explain what async/await does or gives a significantly inaccurate description of asynchronous programming.		Gives a basically correct explanation of async/await and identifies a valid reason to use it, such as avoiding blocking.		Clearly explains async/await, describes Task-based patterns, and gives a concrete example such as an async database call or API request.

Detail

C# Syntax, Data Types, and Control Flow (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.

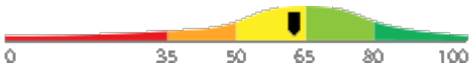
Can you explain what a pointer is in C and describe a situation where you would use one?

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

Object-Oriented Programming (OOP) with Classes and Interfaces (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.

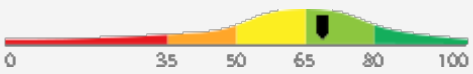
Can you explain what a pointer is in C and describe a situation where you would use one?

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

C# Syntax, Data Types, and Control Flow

Score: 69



Description:

Covers the foundational building blocks of C# programming, including variables, data types, operators, conditional statements, loops, and methods. This is the base layer of knowledge required to write any functional C# code and is applied constantly in everyday programming tasks.

Interpretation:

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

The candidate demonstrates a solid working knowledge of C# programming, including object-oriented principles, exception handling, collections, and data manipulation techniques. Competence in areas such as file I/O, unit testing, database connectivity, and version control is evident, though some advanced topics may benefit from further refinement. This candidate is well-positioned to contribute effectively in an entry-level to mid-level C# development role.

Interview Guide

What is the difference between value types and reference types in C#, and how does that difference affect the way you write and reason about your code day to day?

1	2	3	4	5
Cannot distinguish value from reference types or gives a vague, inaccurate explanation.		Correctly distinguishes the two types but gives limited examples of practical impact on code behavior.		Clearly explains the distinction with concrete examples of memory behavior, null handling, and real coding implications.

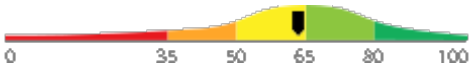
Can you walk me through how you would declare a variable in C#, and give an example of how you might use an if statement and a loop together to solve a simple problem?

1	2	3	4	5
Cannot clearly explain variable declaration or struggles to describe basic if/loop logic.		Correctly describes variable declaration and demonstrates basic if/loop usage with minor gaps.		Clearly explains typed variables, provides a clean, practical example combining if logic and loops confidently.

Detail

Collections, LINQ, and Data Manipulation

Score: 63



Description:

Covers the use of common collection types such as List, Dictionary, and arrays, as well as LINQ queries and lambda expressions for filtering, sorting, and transforming data. These tools are used constantly in business applications to work with sets of data.

Interpretation:

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

The candidate possesses a moderate understanding of common C# collection types and basic LINQ operations, demonstrating familiarity with core concepts. However, gaps in knowledge may limit their ability to confidently handle more complex data filtering, sorting, or transformation tasks independently.

Interview Guide

Can you walk through how you would use LINQ to filter and sort a list of objects in C#? What are some advantages of using LINQ over writing manual loops for data manipulation?

1	2	3	4	5
Cannot write or describe a LINQ query or is unable to articulate any advantage over manual loops.		Correctly demonstrates a basic LINQ query and identifies readability or conciseness as an advantage.		Writes a clear LINQ example with filtering and sorting, and articulates multiple practical advantages including readability, composability, and reduced error risk.

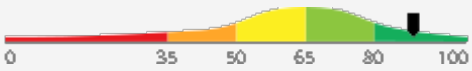
Have you worked with lists or dictionaries in C#? Can you describe how you would store a group of items and then find or filter specific ones from that group?

1	2	3	4	5
Cannot describe basic list or dictionary usage or is unable to explain how to find or filter items.		Describes using a list or dictionary correctly and can explain a basic approach to filtering, even if not using LINQ.		Confidently explains collections and demonstrates knowledge of LINQ or lambda expressions to filter and query data efficiently.

Detail

Database Access with Entity Framework and ADO.NET

Score: 88



Description:

Covers how to connect C# applications to databases using Entity Framework (including LINQ-based queries and model configuration) and ADO.NET for lower-level data access. Most business applications require reading from and writing to a database, making this a high-frequency practical skill.

Interpretation:

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

The candidate demonstrates an advanced and comprehensive mastery of database access in C#, reflecting deep expertise in both Entity Framework and ADO.NET. This individual is highly capable of architecting, implementing, and optimizing database-connected applications, making them a strong asset for roles where data access is a high-frequency and critical responsibility.

Interview Guide

Can you explain the difference between using Entity Framework and ADO.NET to access a database in C#? In what situations might you choose one over the other?

1	2	3	4	5
Cannot distinguish Entity Framework from ADO.NET or gives an inaccurate comparison.		Correctly distinguishes the two at a high level and gives a general reason for choosing one, such as simplicity vs. control.		Clearly explains both tools, their trade-offs, and gives specific, practical scenarios where each is the better choice.

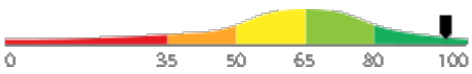
Have you connected a C# application to a database before? Can you describe how you retrieved or saved data, even at a basic level?

1	2	3	4	5
Cannot describe any method of connecting to or querying a database from C#.		Describes a basic approach to database access using either Entity Framework or ADO.NET with some accuracy.		Clearly describes a practical database access approach, names the tools used, and explains how data was retrieved, saved, or updated.

Detail

Exception Handling and Debugging

Score: 95



Description:

Covers how to use try/catch/finally blocks to handle errors gracefully, how to throw and create custom exceptions, and how to use debugging tools and error logging to identify and fix problems in code. These skills are essential for writing reliable, maintainable applications.

Interpretation:

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

The candidate demonstrates an advanced and comprehensive mastery of exception handling and debugging in C#. They are highly proficient in designing robust error-handling strategies, creating custom exceptions, and leveraging debugging tools and error logging to produce reliable, maintainable applications.

Interview Guide

How do you decide what exceptions to catch and handle versus which ones to let bubble up in a C# application? Can you give an example from past experience?

1	2	3	4	5
Cannot articulate a reasoning process for exception handling decisions or gives an impractical answer.		Gives a reasonable general rule for catching exceptions and provides a basic example.		Articulates a clear, principled approach to exception handling with a specific, practical example and mentions logging or re-throwing appropriately.

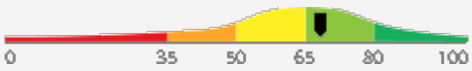
What do you do when your C# code throws an error at runtime? Can you describe how you would use a try/catch block and what steps you take to figure out what went wrong?

1	2	3	4	5
Cannot describe try/catch usage or has no clear process for investigating a runtime error.		Correctly describes try/catch structure and mentions using error messages or a debugger to investigate issues.		Clearly explains try/catch/finally, describes a structured debugging approach, and mentions logging or tools like Visual Studio debugger.

Detail

Object-Oriented Programming (OOP) with Classes and Interfaces

Score: 68



Description:

Covers how to design and use classes, objects, constructors, access modifiers, properties, inheritance, interfaces, and polymorphism in C#. These concepts are central to structuring any non-trivial C# application and are used in virtually every business codebase.

Interpretation:

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

The candidate exhibits a solid and well-rounded understanding of OOP principles in C#, including the effective use of classes, inheritance, interfaces, properties, and polymorphism. They are likely capable of contributing meaningfully to professional C# codebases with minimal supervision, though occasional gaps in advanced design scenarios may be present.

Interview Guide

Can you explain the difference between an abstract class and an interface in C#, and describe a situation where you would choose one over the other?

1	2	3	4	5
Cannot clearly distinguish an abstract class from an interface or gives an inaccurate explanation.		Correctly distinguishes the two but gives a generic or textbook reason for choosing one over the other.		Clearly explains both concepts and gives a specific, practical scenario demonstrating sound design judgment.

Can you explain what a class is in C# and describe how you would create one with a property and a method? Have you worked with inheritance or interfaces at all?

1	2	3	4	5
Struggles to define a class or cannot describe a property or method with any accuracy.		Correctly defines a class with a property and method; has a basic, surface-level understanding of inheritance or interfaces.		Clearly defines and explains class structure, properties, and methods, and gives a meaningful example of inheritance or interface use.

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.

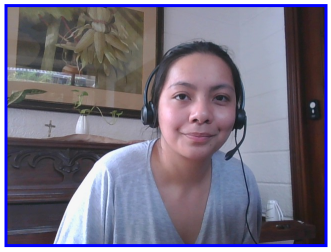
Question or Task	Response

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

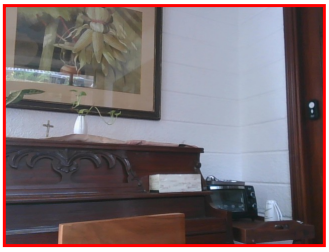
The following photos of the candidate and any identification were uploaded during the assessment session.

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%)

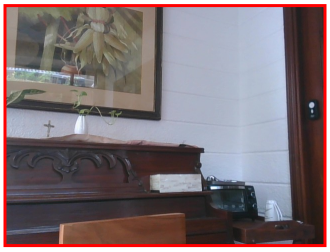


Pre/Post-Test Photo

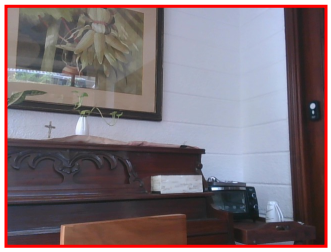
ID Photo



In-Test Error Detected (No Face Detected)

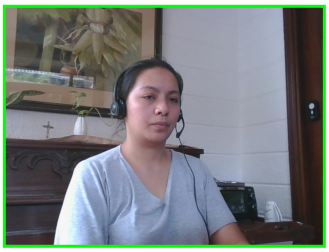


In-Test Error Detected (No Face Detected)

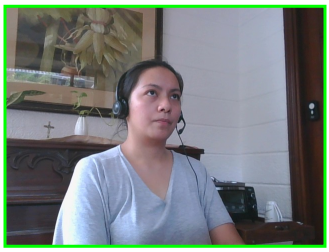


In-Test Error Detected (No Face Detected)

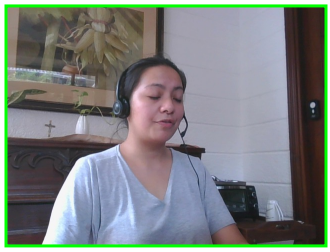
In-Test Photo



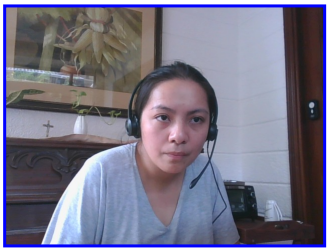
In-Test Photo



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: 20779-1, Key: 0-0, Rpt: 68, Prd: 9601, Created: 2026-06-25 16:20 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 with Async and Await	90.1788	Numeric Score	90.1788	12.5000
C# Syntax, Data Types, and Control Flow	69.0802	Numeric Score	69.0802	12.5000
C# Syntax, Data Types, and Control Flow (Coding Tasks)	62.9784	Numeric Score	62.9784	12.5000
Collections, LINQ, and Data Manipulation	63.7559	Numeric Score	63.7559	12.5000
Database Access with Entity Framework and ADO.NET	88.7213	Numeric Score	88.7213	12.5000
Exception Handling and Debugging	95.7526	Numeric Score	95.7526	12.5000
Object-Oriented Programming (OOP) with Classes and Interfaces	68.5788	Numeric Score	68.5788	12.5000
Object-Oriented Programming (OOP) with Classes and Interfaces (Coding Tasks)	62.9784	Numeric Score	62.9784	12.5000
Weighted Average:				75.2531
Final Overall Score:				75

Notes

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