hr | avatar
Illuminating Talent

Test Results and Interview Guide

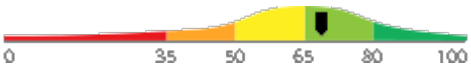
Candidate: **Elizabeth Wantsajob**
Assessment: Kotlin
Completed: June 28, 2026
Prepared for: Sara Maple
Example Company

What's Included

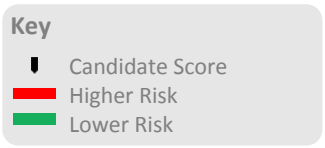
- Overall Score
- Competency Summary Table
- Comparison Matrix
- Detailed Competency Results with Interview Guide

Important Note: The Kotlin 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.

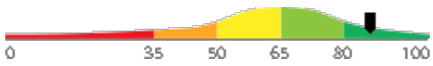
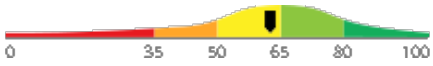
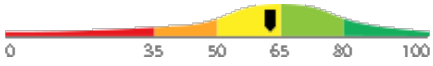
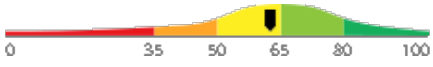
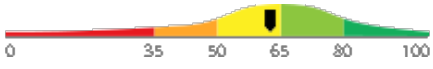
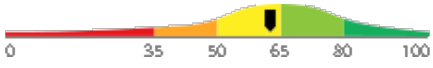
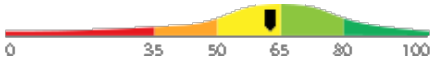
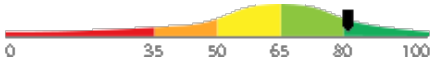
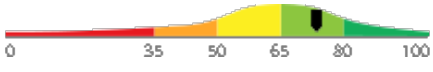
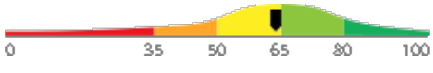
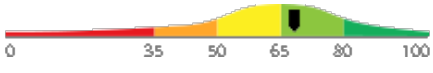
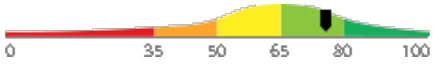
Overall

Candidate	Score	Interpretation
Elizabeth Wantsajob beth.wantsajob@gmail.com Kotlin June 28, 2026	69	

The candidate demonstrates a solid and well-rounded understanding of Kotlin and Android development, including competence in syntax, null safety, collections, lifecycle management, navigation, and common architecture patterns. While some gaps may exist in specialized or advanced areas such as complex coroutine usage, advanced Gradle configuration, or nuanced debugging techniques, the candidate is generally well-prepared to contribute effectively to Android development projects with moderate supervision.



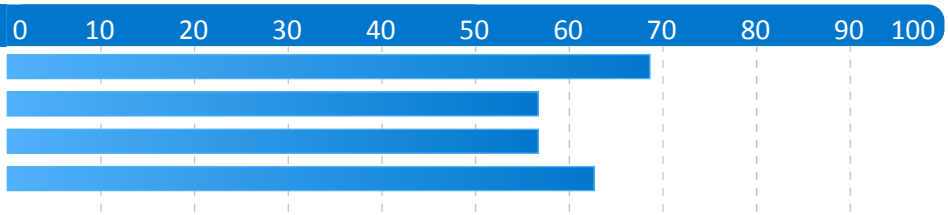
Competency Summary

Competency	Score	Interpretation
Skills/Knowledge (relates to immediate readiness)		
Android Architecture Components (ViewModel, LiveData, and Room)	86	
Android Architecture Components (ViewModel, LiveData, and Room) (Coding Tasks)	62	
Android UI, Navigation, and User Input (Coding Tasks)	62	
Collections, Lambdas, and Extension Functions (Coding Tasks)	62	
Coroutines and Background Threading (Coding Tasks)	62	
Kotlin Syntax and Core Language Features (Coding Tasks)	62	
Null Safety (Coding Tasks)	62	
Android UI, Navigation, and User Input	81	
Collections, Lambdas, and Extension Functions	74	
Coroutines and Background Threading	64	
Kotlin Syntax and Core Language Features	68	
Null Safety	76	

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	69th
North America	57th
United States	57th
Example Company	63rd



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:	Kotlin
Authorized:	June 28, 2026, by Sara Maple, Example Company, qamailsaram.mike@hravatar.com
Started:	June 28, 2026, 1:08:59PM EDT
Completed:	June 28, 2026, 1:08:59PM EDT
Overall Score:	69

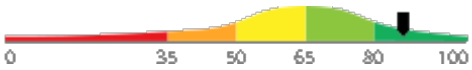
Knowledge and Skills Detail

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

Android Architecture Components (ViewModel, LiveData, and Room)

Score: 86



Description:

Covers the use of Android Jetpack architecture components including ViewModel for managing UI-related data, LiveData for observing data changes in a lifecycle-aware way, and Room for local data persistence. These components are central to building well-structured, maintainable Android apps.

Interpretation:

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

The candidate exhibits an advanced and comprehensive mastery of Android Jetpack architecture components, reflecting deep expertise in leveraging ViewModel, LiveData, and Room to build well-structured, lifecycle-aware, and data-persistent Android applications. They are well-equipped to lead development efforts, mentor others, and architect robust solutions using these components.

Describe how you would set up a Room database in an Android app to store and retrieve a simple list of items, and explain the key components involved.

1	2	3	4	5
Cannot describe Room setup or confuses it with other storage options like SharedPreferences.		Identifies the Entity, DAO, and Database classes and describes a basic setup for storing and querying data.		Accurately describes all three components, mentions using coroutines or LiveData with DAO queries, and discusses database migration or type converters.

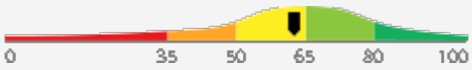
What is a ViewModel in Android, and why would you use one instead of storing data directly in an Activity or Fragment?

1	2	3	4	5
Cannot explain what a ViewModel is or does not understand why it is needed.		Explains that ViewModel survives configuration changes and keeps UI data separate from UI controllers.		Explains lifecycle awareness, data survival across rotation, separation of concerns, and how it works with LiveData or StateFlow.

Detail

Android Architecture Components (ViewModel, LiveData, and Room) (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.

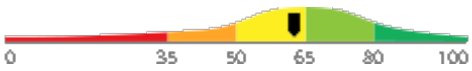
Can you explain what a pointer is in C and describe a situation where you would use one?

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

Android UI, Navigation, and User Input (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.

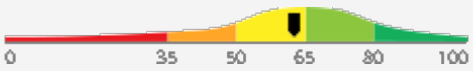
Can you explain what a pointer is in C and describe a situation where you would use one?

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

Collections, Lambdas, and Extension Functions (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.

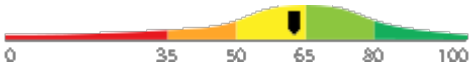
Can you explain what a pointer is in C and describe a situation where you would use one?

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

Coroutines and Background Threading (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.

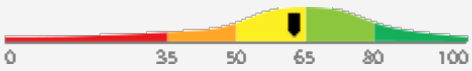
Can you explain what a pointer is in C and describe a situation where you would use one?

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

Kotlin Syntax and Core Language Features (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.

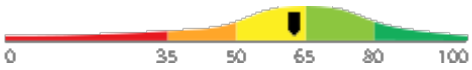
Can you explain what a pointer is in C and describe a situation where you would use one?

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

Null Safety (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.

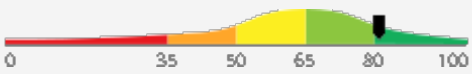
Can you explain what a pointer is in C and describe a situation where you would use one?

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

Android UI, Navigation, and User Input

Score: 81



Description:

Covers building and managing Android user interfaces using XML layouts and Kotlin, handling user input and events, passing data between screens using intents or navigation components, and managing the Activity and Fragment lifecycle. These skills are required for nearly every Android feature a developer builds.

Interpretation:

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

The candidate exhibits an advanced and comprehensive understanding of Android UI development, navigation, and user input handling in Kotlin. They demonstrate strong proficiency in XML layouts, lifecycle management, event handling, and data passing via intents and navigation components, reflecting the depth of skill required for professional Android feature development.

Interview Guide

Explain the Android Activity lifecycle and describe what code you would put in onCreate, onPause, and onDestroy, and why.

1	2	3	4	5
Cannot name the lifecycle methods in order or does not know what logic belongs in each callback.		Correctly names the key lifecycle methods and provides reasonable examples of what to initialize, pause, or release in each.		Provides accurate and detailed examples for each method, explains why misplacing logic causes bugs or memory leaks, and mentions Fragment lifecycle differences.

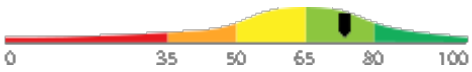
How would you respond to a button click in an Android app using Kotlin? Can you describe or write the code you would use?

1	2	3	4	5
Cannot describe how to set a click listener or confuses the method names and syntax.		Correctly describes setting an OnClickListener on a button using Kotlin, either with a lambda or anonymous class.		Demonstrates clean Kotlin lambda syntax, mentions view binding as a preferred approach over findViewById, and explains why view binding reduces errors.

Detail

Collections, Lambdas, and Extension Functions

Score: 74



Description:

Covers working with Kotlin's standard collection types such as lists, maps, and sets, and using higher-order functions and lambdas to filter, transform, and process data. Also includes writing and using extension functions to add functionality to existing classes in a clean and reusable way.

Interpretation:

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

The candidate shows a solid and proficient understanding of Kotlin collections, higher-order functions, lambdas, and extension functions. They are capable of effectively filtering, transforming, and processing data, as well as writing clean and reusable extension functions with only occasional gaps in advanced application.

Interview Guide

What is an extension function in Kotlin, and can you give an example of one you have written or would write to solve a practical problem in an Android app?

1	2	3	4	5
Cannot define extension functions or confuses them with regular class methods.		Correctly defines extension functions and provides a simple, practical example such as a String or View extension.		Provides a well-reasoned example, explains how extension functions improve code readability, and notes their limitations such as no access to private members.

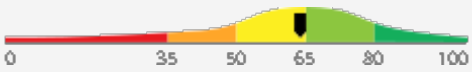
How would you use Kotlin to filter a list of numbers and return only the ones greater than 10? Can you write or describe the code you would use?

1	2	3	4	5
Cannot describe how to filter a list or attempts to use a manual loop without knowledge of collection functions.		Correctly uses the filter function with a lambda expression to return numbers greater than 10.		Uses filter fluently, mentions chaining with other functions like map or forEach, and explains lambda syntax clearly.

Detail

Coroutines and Background Threading

Score: 64



Description:

Covers the use of Kotlin coroutines to perform tasks on background threads, such as network calls and database operations, without blocking the main UI thread. Includes understanding of coroutine scopes, suspend functions, and dispatchers commonly used in Android development.

Interpretation:

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

The candidate possesses a moderate understanding of Kotlin coroutines and background threading. They demonstrate a working knowledge of coroutine scopes, suspend functions, and dispatchers, though gaps in understanding may limit their ability to handle more complex or nuanced background threading scenarios.

Interview Guide

Walk me through how you would use a coroutine inside a ViewModel to fetch data from a network and update the UI when the result is ready.

1	2	3	4	5
Cannot describe coroutine usage in a ViewModel or does not know how to update the UI from a coroutine result.		Describes launching a coroutine in viewModelScope, calling a suspend function, and posting the result to LiveData or a state variable.		Provides a complete and accurate flow including viewModelScope, Dispatchers.IO for the network call, switching context, error handling with try-catch, and updating UI state.

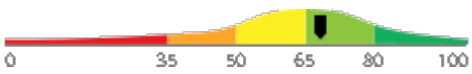
Why is it a problem to run a network request on the main thread in Android, and what is one way Kotlin helps you avoid this problem?

1	2	3	4	5
Cannot explain why blocking the main thread is harmful or does not know how to move work off it.		Explains that blocking the main thread causes ANR errors and mentions coroutines or background threads as a solution.		Clearly explains the ANR risk, describes how coroutines with the appropriate dispatcher move work off the main thread, and mentions suspend functions.

Detail

Kotlin Syntax and Core Language Features

Score: 68



Description:

Covers the foundational building blocks of Kotlin, including variables, data types, functions, classes, interfaces, and inheritance. Also includes control flow structures such as loops, conditionals, and when expressions, which are used constantly in everyday Android development.

Interpretation:

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

The candidate exhibits a solid working knowledge of Kotlin syntax, Android application development, and many associated tools and frameworks, including null safety, collections, lifecycle management, and basic architecture components. They are likely capable of contributing effectively to Android development tasks at an entry-level to mid-level capacity with minimal supervision.

Interview Guide

Walk me through how you would use a 'when' expression in Kotlin to handle multiple conditions, and explain how it differs from a traditional 'if-else' chain.

1	2	3	4	5
Cannot describe 'when' syntax or incorrectly equates it to a simple switch statement.		Correctly describes 'when' syntax and notes it can replace if-else chains with cleaner code.		Demonstrates advanced uses such as argument-less when, range checks, and using when as an expression returning a value.

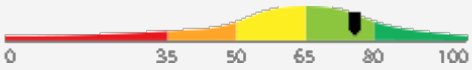
Can you explain the difference between 'val' and 'var' in Kotlin, and give an example of when you would use each one?

1	2	3	4	5
Cannot distinguish val from var or provides incorrect examples.		Correctly defines val as immutable and var as mutable with a basic example.		Clearly explains both, discusses practical implications, and mentions compile-time enforcement.

Detail

Null Safety

Score: 76



Description:

Covers Kotlin's built-in system for preventing null pointer exceptions, including nullable types, safe call operators (?.), the Elvis operator (?:), and non-null assertions (!!). Null safety is a core part of writing correct and crash-free Kotlin code.

Interpretation:

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

The candidate demonstrates a solid understanding of Kotlin's null safety features, including nullable types, safe call operators, the Elvis operator, and non-null assertions. They are generally capable of applying these concepts to write stable and crash-resistant Kotlin code, with minor areas for continued growth.

Interview Guide

Describe a real situation in Android development where you encountered a null value and explain how you used Kotlin's null safety features to handle it correctly.

1	2	3	4	5
Cannot describe a concrete scenario or relies solely on non-null assertions without explanation.		Describes a plausible scenario and correctly applies safe calls or the Elvis operator to handle null.		Provides a detailed scenario, explains the choice of null safety tool used, and discusses trade-offs of different approaches.

What does it mean for a variable to be nullable in Kotlin, and how would you safely access a property on a variable that might be null?

1	2	3	4	5
Cannot explain nullable types or does not know how to safely access nullable properties.		Explains nullable types using '?' and demonstrates safe call operator with a basic example.		Covers safe calls, Elvis operator, and explains when to avoid the non-null assertion operator (!!).

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.

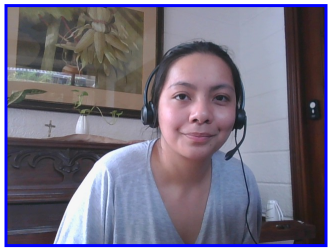
Question or Task	Response

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

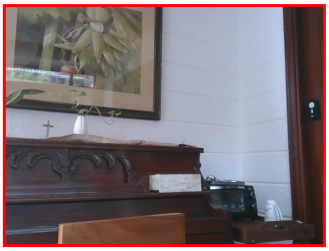
The following photos of the candidate and any identification were uploaded during the assessment session.

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%)

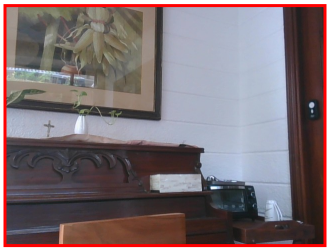


Pre/Post-Test Photo

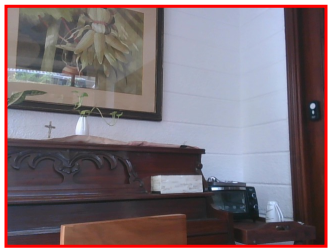
ID Photo



In-Test Error Detected (No Face Detected)

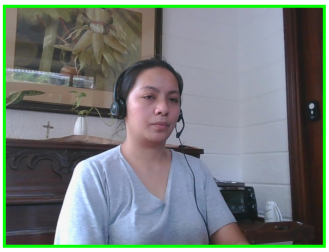


In-Test Error Detected (No Face Detected)

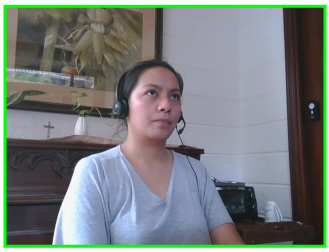


In-Test Error Detected (No Face Detected)

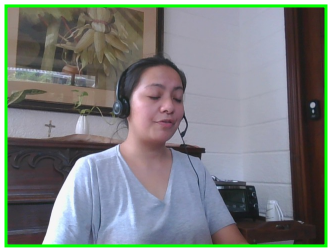
In-Test Photo



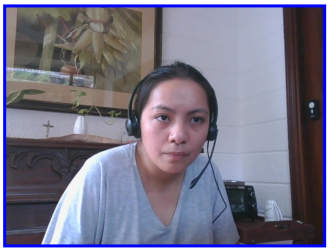
In-Test Photo



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: 20816-1, Key: 0-0, Rpt: 68, Prd: 9638, Created: 2026-06-28 13:09 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 (%)
Android Architecture Components (ViewModel, LiveData, and Room)	86.5219	Numeric Score	86.5219	8.3333
Android Architecture Components (ViewModel, LiveData, and Room) (Coding Tasks)	62.9784	Numeric Score	62.9784	8.3333
Android UI, Navigation, and User Input	81.3409	Numeric Score	81.3409	8.3333
Android UI, Navigation, and User Input (Coding Tasks)	62.9784	Numeric Score	62.9784	8.3333
Collections, Lambdas, and Extension Functions	74.1170	Numeric Score	74.1170	8.3333
Collections, Lambdas, and Extension Functions (Coding Tasks)	62.9784	Numeric Score	62.9784	8.3333
Coroutines and Background Threading	64.1878	Numeric Score	64.1878	8.3333
Coroutines and Background Threading (Coding Tasks)	62.9784	Numeric Score	62.9784	8.3333
Kotlin Syntax and Core Language Features	68.6402	Numeric Score	68.6402	8.3333
Kotlin Syntax and Core Language Features (Coding Tasks)	62.9784	Numeric Score	62.9784	8.3333
Null Safety	76.1602	Numeric Score	76.1602	8.3333
Null Safety (Coding Tasks)	62.9784	Numeric Score	62.9784	8.3333
Weighted Average:				69.0699
Final Overall Score:				69

Notes

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