**Pre-interview test answers**

1. Yes, I stay in Benin City, Edo state.

2. I am very much comfortable working full-time onsite (in the office at Edo Innovation Hub) every day.

3. **Part A:** Analyzing the Pseudocode:

Although there are a few errors in the provided pseudocode, I think the author was aiming to accomplish two main goals:

**Goal 1:** **Populate an Array**: The first part of the pseudocode is designed to create an array x of size 100 and populate it with values. Each element x[a] in the array is meant to store the value 10 + a, where a is the index ranging from 0 to 99. This results in the array containing values from 10 to 109. So, for x[0], the value would be 10, for x[1], the value would be 11, and so on up to x[99] = 109.

**Goal 2: Print Specific Elements of the Array:** The second part aims to print elements of the array x where the index p is divisible by 3 or is a multiple of 3. As p increases by 3, the code is designed to check if p is divisible by 3 and print the corresponding element of x[p].

**Errors identified:**

* The loop does not increase the counter a, so it runs forever (infinite loop). There is the need to add a = a + 1 inside the first loop.
* The condition While p > 100 prevents the second loop from running, as p starts at 1, which is never greater than 100. The correct condition should be p < 100.
* The code checks if p is divisible by 3 in a complex way. It can be simplified to p % 3 = 0, Implementing the use of the modulus ( % ) operator for efficiency.
* p starts at 1, the condition p % 3 == 0 will never be true since p is initialized to 1 and then increased by 3 on each iteration. As a result, the print x[p] will always be unreachable because p will always take values like 4, 7, 10, etc., which are not divisible by 3.

3**. PART B: A better version of the code.**

# Goal 1: Populate the array

Create array x[100]

Create integer a = 0

While a < 100

x[a] = 10 + a

a = a + 1 # Properly increment a

End while

# Goal 2: Print elements at indices divisible by 3

Create integer p = 0 # Initialize p to 0

While p < 100

p = p + 3 # Properly increment p

if p % 3 = 0 then # Implement the use of modulus operator instead

print x[p]

end if

End while

**Why I wrote the improved code:**

1. **Correctness and Functionality:** Several logical mistakes in the original pseudocode prevented it from functioning as planned. The code can now correctly fill the array and output particular numbers by changing the loop conditions and adding the appropriate incrementation (a = a + 1 in the first loop and p = p + 3 in the second loop). Making sure the loops end appropriately is essential to the program's functionality and helps avoid infinite loops.
2. **Efficiency and Simplicity:** In order to make things simpler, I introduced the modulus operator (p % 3 = 0), which is a common and effective method of determining if a given number is divisible by another. This lowers computational overhead while also making the code easier to understand.
3. Improved Logic in Handling Arrays: We start at the proper place for divisibility by 3 and make sure that all valid indices of the array—0, 3, 6, etc.—are taken into account by initializing p to 0. This is significant because, if one were to start at p = 1, significant array indices would be missed, missing values that the original author probably intended to include. The code is now logically consistent with the objective of printing elements where the index is divisible by three thanks to this minor change.

**4.** Aquery to show the longest book written by each author:

SELECT

Author\_Name,

Book\_Name,

Number\_Of\_Pages

FROM

Books AS b

WHERE

Number\_Of\_Pages = (

SELECT MAX(Number\_Of\_Pages)

FROM

Books AS sub

WHERE

sub.Author\_Name = b.Author\_Name

)

ORDER BY

Author\_Name, Book\_Name;

**Here’s a concise explanation of the solution:**  
This query retrieves the longest book for each author. It works by selecting Author\_Name, Book\_Name, and Number\_Of\_Pages from the Books table. For each book, the **correlated subquery** runs and finds the maximum number of pages (MAX(Number\_Of\_Pages)) for that specific author. The outer query then filters to include only the books where Number\_Of\_Pages matches this maximum. The results are ordered by Author\_Name and Book\_Name for better presentation.

The **correlated subquery** is crucial as it dynamically compares each author's book against their longest one, making the solution both efficient and precise.