Accenture Sections	Information	Questions and Time
Cognitive Ability	• English Ability • Critical Thinking and Problem Solving • Abstract Reasoning	50 Ques in 50 mins
Technical Assessment	• Common Application and MS Office • Pseudo Code • Fundamental of Networking, Security and Cloud	40 Ques in 40 mins
Coding Round	• C • C++ • Dot Net • JAVA • Python	2 Ques in 45 mins

DEBUG WITH SHUBHAM

Accenture Technical Assessment Detailed Overview

Pseudo Code

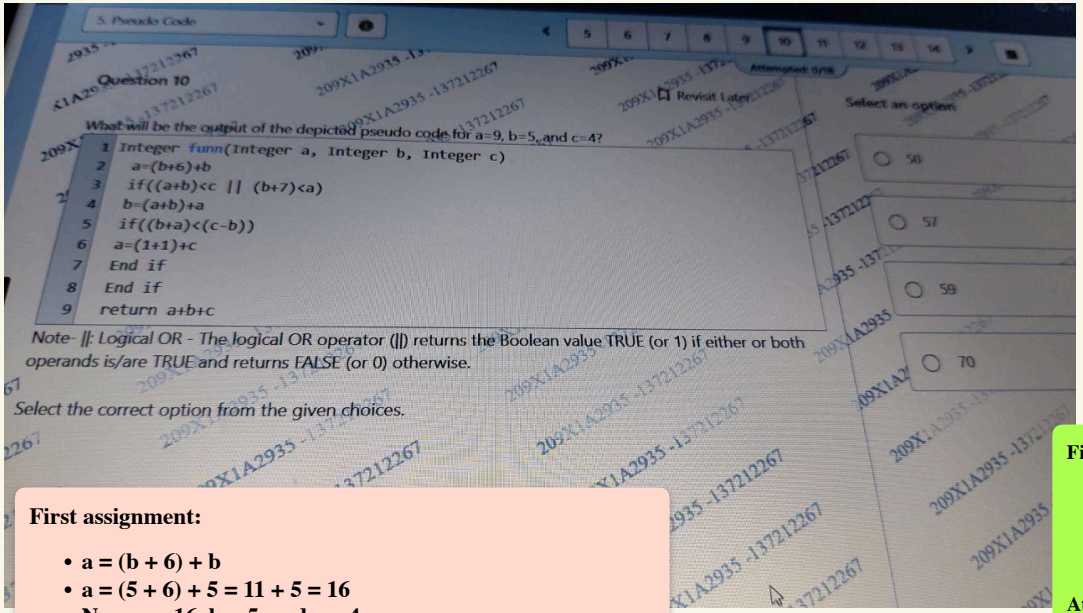

https://www.youtube.com/@DebugWithShubham

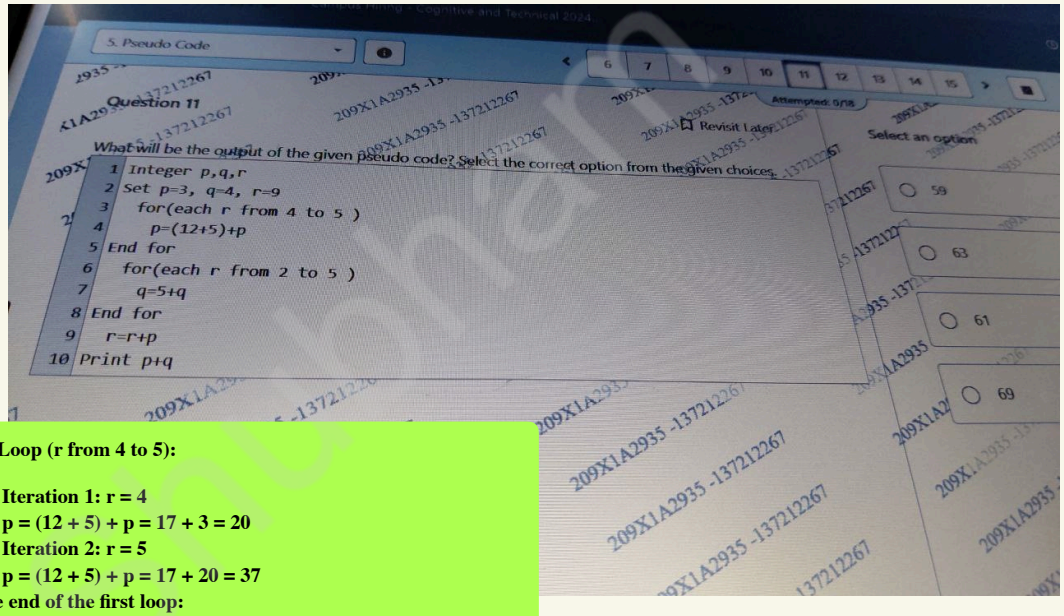

https://www.linkedin.com/in/debugwithshubham/


https://www.instagram.com/debugwithshubham/


https://topmate.io/debugwithshubham


https://t.me/debugwithshubham





First assignment:

- **a = (b + 6) + b**
- **a = (5 + 6) + 5 = 11 + 5 = 16**
- **Now, a = 16, b = 5, and c = 4.**

First condition check:

- **if ((a + b) < c || (b + 7) < a)**
- **Substituting the values: if ((16 + 5) < 4 || (5 + 7) < 16)**
- **if (21 < 4 || 12 < 16) → false || true → true**
- **Since this condition is true, we proceed into the if block.**

Assignment inside the first if block:

- **b = (a + b) + a**
- **b = (16 + 5) + 16 = 21 + 16 = 37**
- **Now, b = 37, a = 16, and c = 4.**

Second condition check:

- **if ((b + a) < (c - b))**
- **Substituting the values: if ((37 + 16) < (4 - 37))**
- **if (53 < -33) → false**
- **Since this condition is false, the second if block is skipped.**

Return the result:

- **The function returns a + b + c**
- **a + b + c = 16 + 37 + 4 = 57**

First Loop (r from 4 to 5):

- **Iteration 1: r = 4**
  - **p = (12 + 5) + p = 17 + 3 = 20**
- **Iteration 2: r = 5**
  - **p = (12 + 5) + p = 17 + 20 = 37**

At the end of the first loop:

- **p = 37**

Second Loop (r from 2 to 5):

- **Iteration 1: r = 2**
  - **q = 5 + q = 5 + 4 = 9**
- **Iteration 2: r = 3**
  - **q = 5 + q = 5 + 9 = 14**
- **Iteration 3: r = 4**
  - **q = 5 + q = 5 + 14 = 19**
- **Iteration 4: r = 5**
  - **q = 5 + q = 5 + 19 = 24**

At the end of the second loop:

- **q = 24**

After the Loops:

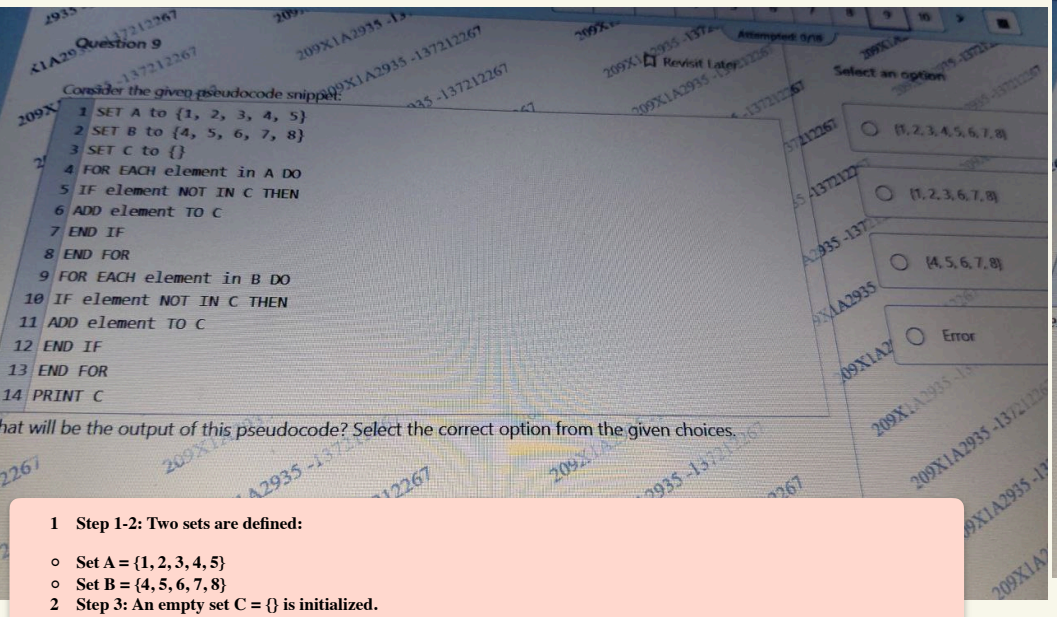
- **r = r + p = 5 + 37 = 42**

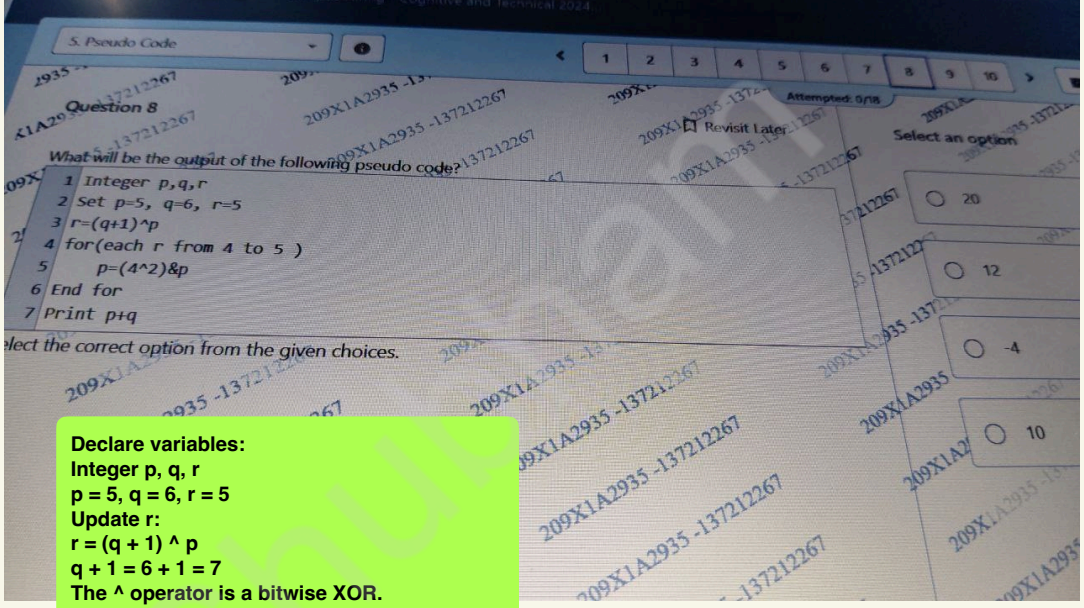
Final Calculation:

- **p + q = 37 + 24 = 61**

Correct Option:

- **The correct option is c) 61.**





1 **Step 1-2: Two sets are defined:**

- **Set A = {1, 2, 3, 4, 5}**
- **Set B = {4, 5, 6, 7, 8}**

2 **Step 3: An empty set C = {} is initialized.**

3 **Step 4-8: A FOR loop runs through each element in set A, and if the element is not already in set C, it is added to C.**

- **After processing set A, set C = {1, 2, 3, 4, 5}.**

4 **Step 9-13: Another FOR loop runs through each element in set B, and if the element is not already in set C, it is added to C.**

- **Set B = {4, 5, 6, 7, 8}. Since 4 and 5 are already in C, only 6, 7, and 8 are added.**
- **After processing set B, set C = {1, 2, 3, 4, 5, 6, 7, 8}.**

5 **Step 14: The program prints set C.**

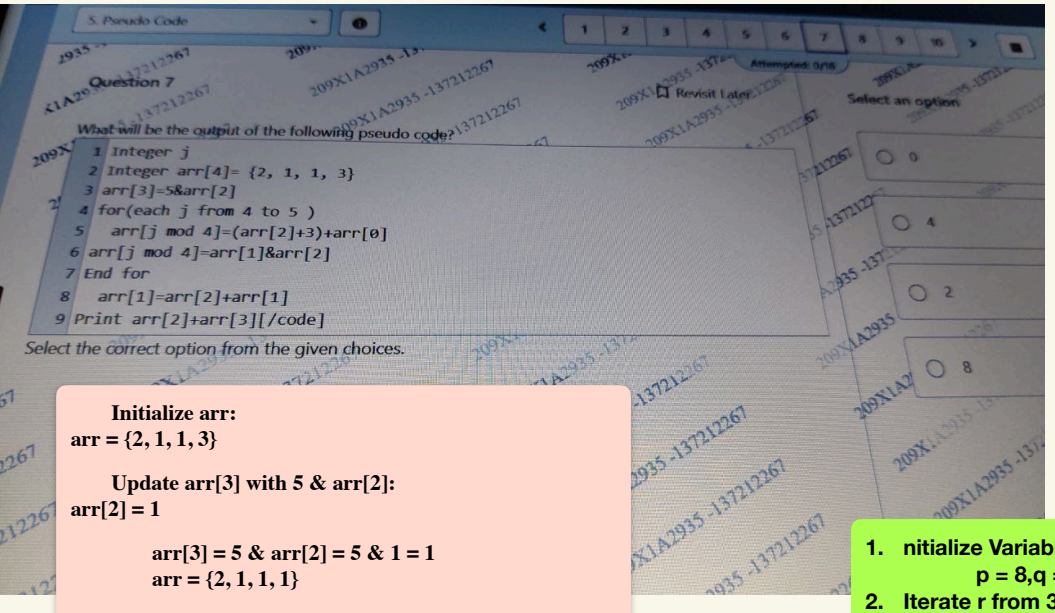
Final Output:

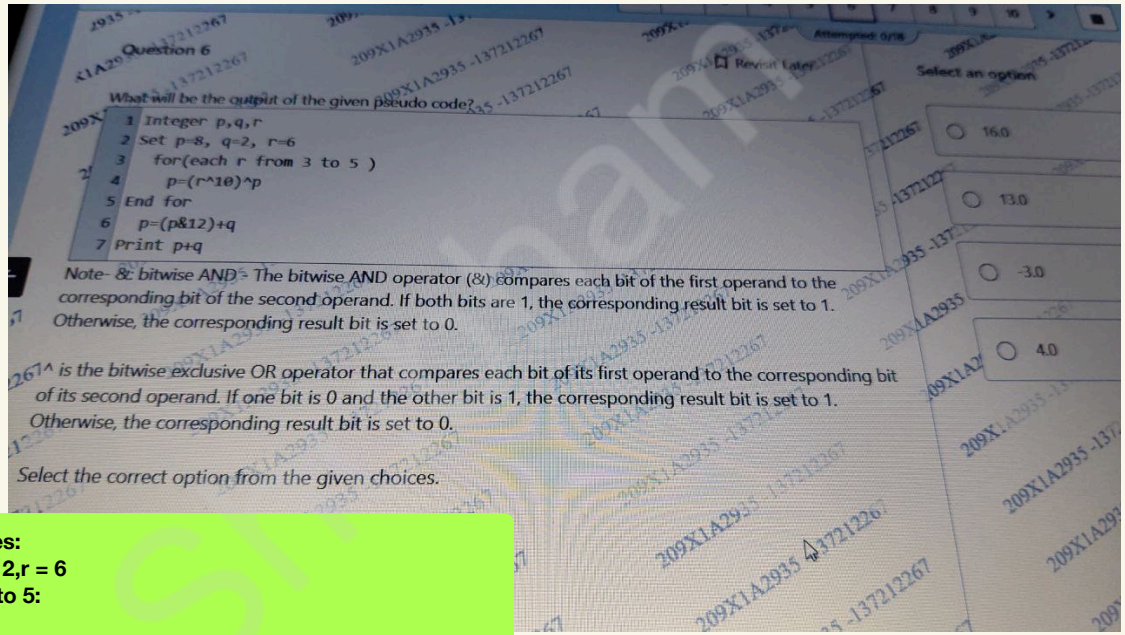
The final set C will be: {1, 2, 3, 4, 5, 6, 7, 8}.

Correct Option:

- **(1, 2, 3, 4, 5, 6, 7, 8)**

Declare variables:
Integer p, q, r
p = 5, q = 6, r = 5
Update r:
r = (q + 1) ^ p
q + 1 = 6 + 1 = 7
The ^ operator is a bitwise XOR.
r = 7 ^ 5
Binary of 7: 0111
Binary of 5: 0101
0111 ^ 0101 = 0010, which is 2
So, r = 2.
Loop for r from 4 to 5:
This loop iterates for r = 4 and r = 5.
First Iteration (r = 4):
p = (4 ^ 2) & p
4 ^ 2 = 6 (binary: 0100 ^ 0010 = 0110)
6 & p = 6 & 5 = 4 (binary: 0110 & 0101 = 0100)
Now, p = 4.
Second Iteration (r = 5):
Again, p = (4 ^ 2) & p
4 ^ 2 is still 6
6 & p = 6 & 4 = 4 (binary: 0110 & 0100 = 0100)
So, p remains 4.
Print p + q:
At this point, p = 4 and q = 6.
p + q = 4 + 6 = 10.





Initialize arr:
arr = {2, 1, 1, 3}

Update arr[3] with 5 & arr[2]:
arr[2] = 1

arr[3] = 5 & arr[2] = 5 & 1 = 1
arr = {2, 1, 1, 1}

Iterate j from 4 to 5:

For j = 4:
j mod 4 = 4 mod 4 = 0

arr[0] = (arr[2] + 3) + arr[0] = (1 + 3) + 2 = 6
arr[0] = 6
arr = {6, 1, 1, 1}
arr[0] = arr[1] & arr[2] = 1 & 1 = 1
arr = {1, 1, 1, 1}

For j = 5:
j mod 4 = 5 mod 4 = 1

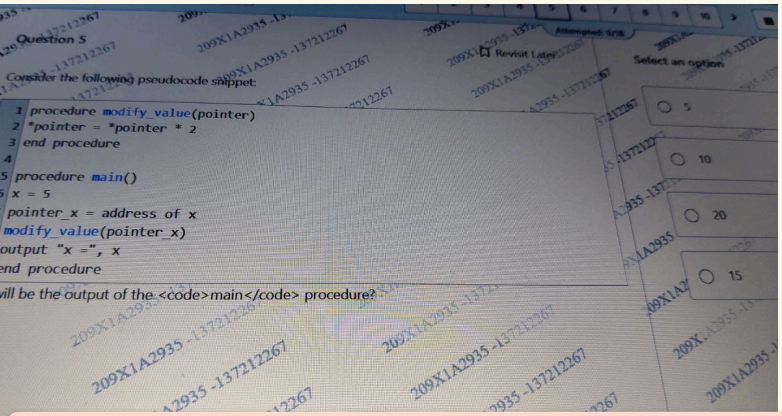
arr[1] = (arr[2] + 3) + arr[0] = (1 + 3) + 1 = 5
arr[1] = 5
arr = {1, 5, 1, 1}
arr[1] = arr[1] & arr[2] = 5 & 1 = 1
arr = {1, 1, 1, 1}
Update arr[1]:
arr[1] = arr[2] + arr[1] = 1 + 1 = 2

arr = {1, 2, 1, 1}
4. Print arr[2] + arr[3]:
arr[2] + arr[3] = 1 + 1 = 2

The final output of the print statement is 2.

1. nitialize Variables:
p = 8,q = 2,r = 6
2. Iterate r from 3 to 5:
For r = 3:
p = (3 ^ 10) ^ p
Calculate 3 ^ 10 using bitwise XOR:
3 (0011 in binary) XOR 10 (1010 in binary) = 1001 (9 in decimal)
9 ^ 8 = 1001 (9) XOR 1000 (8) = 0001 (1 in decimal)
p = 1
For r = 4:
p = (4 ^ 10) ^ p
Calculate 4 ^ 10:
4 (0100 in binary) XOR 10 (1010 in binary) = 1110 (14 in decimal)
14 ^ 1 = 1110 (14) XOR 0001 (1) = 1111 (15 in decimal)
p = 15
For r = 5:
p = (5 ^ 10) ^ p
Calculate 5 ^ 10:
5 (0101 in binary) XOR 10 (1010 in binary) = 1111 (15 in decimal)
15 ^ 15 = 1111 (15) XOR 1111 (15) = 0000 (0 in decimal)
p = 0
Update p:
p = (p & 12) + q

Calculate p & 12:
p = 0 (0000 in binary)
12 = 0000 1100 (in binary)
. p & 12 = 0000 (0 in decimal.
p = 0 + 2 = 2
4. Print p + q:
p + q = 2 + 2 = 4
The final output of the print statement is 4.



1. **Initialization:**
   x = 5

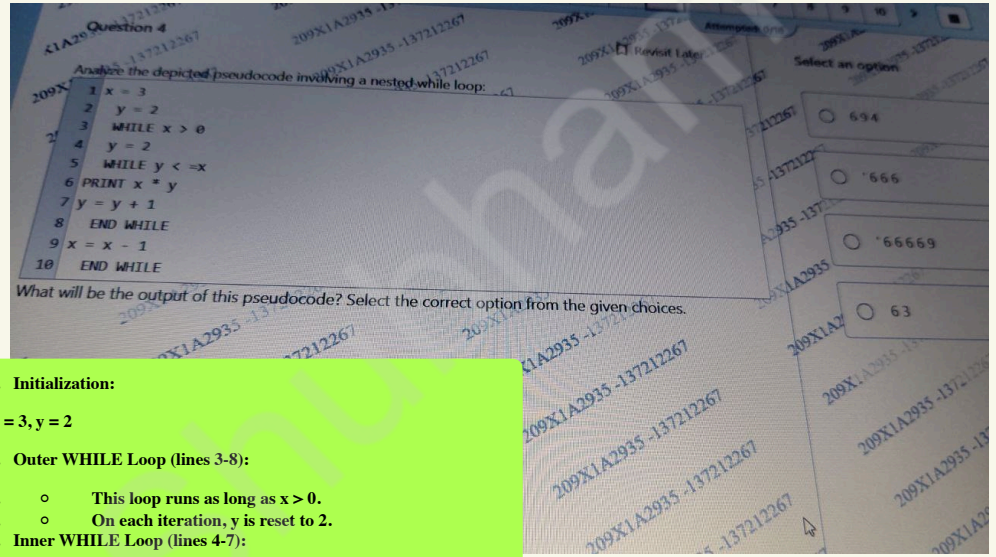
2. **Pointer Assignment:**
   pointer_ = address of x

3. **Call to modify_value(pointer_):**

4. **○ modify_value receives pointer_, which points to x.**
5. **○ Inside modify_value, *pointer is x, so *pointer = *pointer * 2 updates x to 5 * 2 = 10.**
6. **Print Statement:**
   output "x =", 10

The corrected output of the main procedure is:

x = 10



1. **Initialization:**

X = 3, y = 2

1. **Outer WHILE Loop (lines 3-8):**

2. **○ This loop runs as long as x > 0.**
3. **○ On each iteration, y is reset to 2.**
4. **Inner WHILE Loop (lines 4-7):**

5. **○ This loop runs while y <= x.**
6. **○ Inside this loop, the product of x and y is printed.**
7. **○ y is incremented by 1 in each iteration.**
8. **Update x (line 8):**

9. **○ After the inner loop finishes, x is decremented by 1.**

Execution Flow:

Initial values: x = 3,y = 2

Outer Loop Iterations:

- **First Iteration (x = 3):**

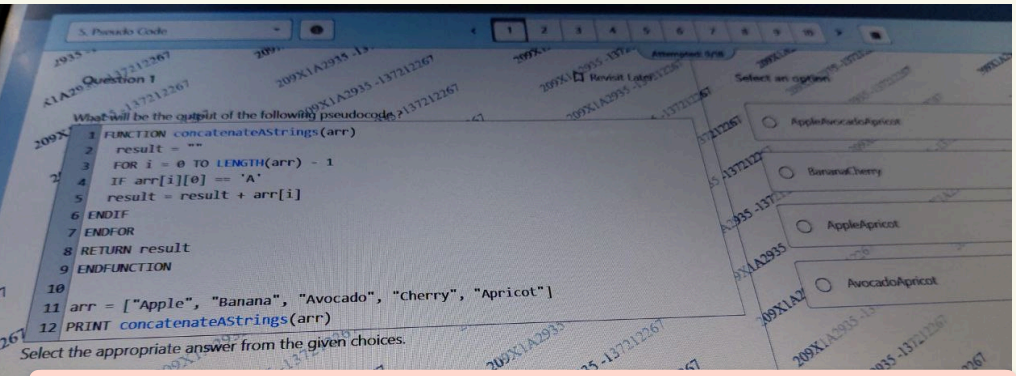
  - **Inner Loop: y starts at 2 and increments while y <= 3.**
    - **Prints 3 * 2 (6)**
    - **Increments y to 3, then prints 3 * 3 (9)**
    - **Increments y to 4, which is no longer <= 3, so the loop ends.**
- **Second Iteration (x = 2):**

  - **Inner Loop: y is reset to 2 and increments while y <= 2.**
    - **Prints 2 * 2 (4)**
    - **Increments y to 3, which is no longer <= 2, so the loop ends.**
- **Third Iteration (x = 1):**

  - **Inner Loop: y is reset to 2 and increments while y <= 1.**
    - **Since y (2) is not <= 1, the inner loop does not execute.**
- **After the third iteration, x is decremented to 0 and the outer loop ends.**

Summary of Outputs:

- **The output : 6 9 4**



Function Definition (concatenateAstrings):

- **Input**: `arr`, an array of strings.

• Output: A concatenated string of elements that start with 'A'.
Initialization:

result = ""
Loop Through Array:

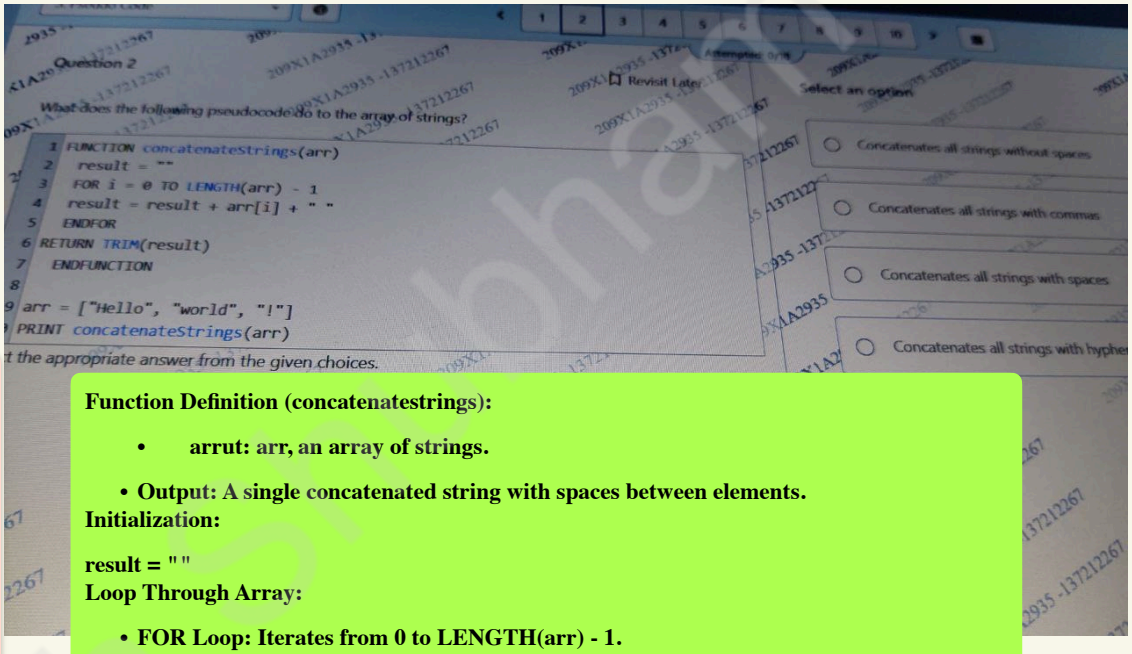
• FOR Loop: Iterates from 0 to LENGTH(arr) - 1.
• Condition: Checks if the first character of arr[i] is 'A'.
- **arr[i][0] = 'A'**

Concatenate Matching Strings:

• If the condition is true, append the string arr[i] to result.
Return Result:

• After processing all elements, return result.
Main Execution:

• arr is defined with the values ["Apple", "Banana", "Avocado", "cherry", "Apricot"].
• concatenateAstrings(arr) is called, and its result is printed.



Function Definition (concatenatestrings):

- **arrut: arr, an array of strings.**

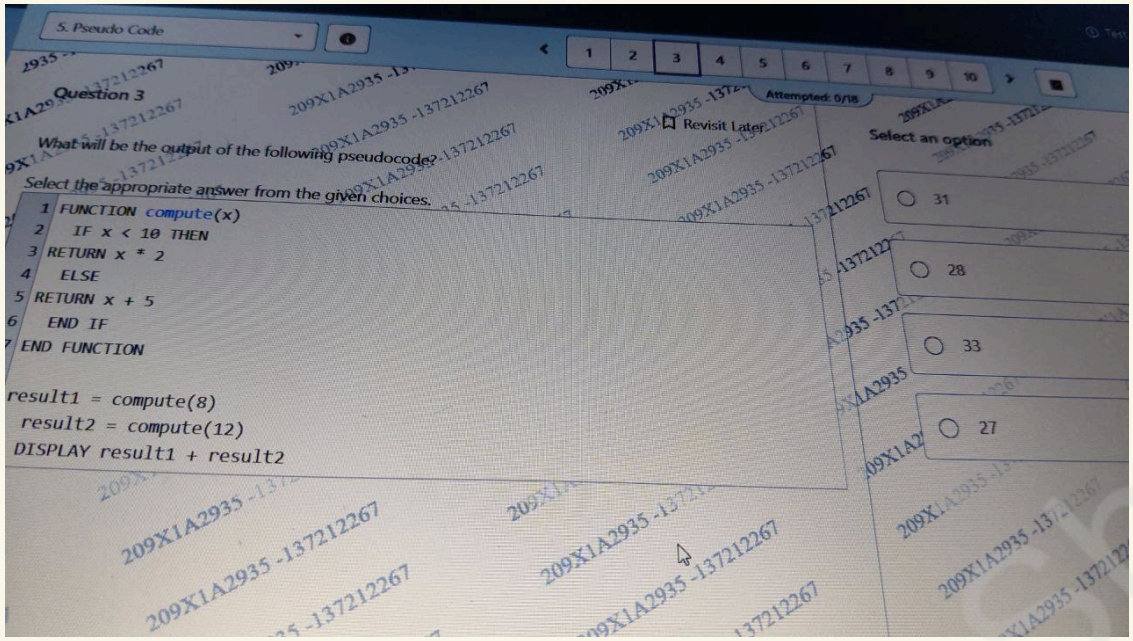
• Output: A single concatenated string with spaces between elements.
Initialization:

result = ""
Loop Through Array:

• FOR Loop: Iterates from 0 to LENGTH(arr) - 1.
• Concatenation: Each string arr[i] is concatenated to result with an additional space " ".
Trim Result:

• TRIM(result) is used to remove any extra space at the end of the concatenated string.
Main Execution:

• arr is defined with the values ["Hello", "world", "!"].
• 3) Concatenate all strings with space result is printed.



Function Definition (compute(x)):

- **Input: x, a number.**
- **Output:**
  - **If x is less than 10, the function returns x * 2.**
  - **Otherwise, it returns x + 5.**

Function Calls:

- **result1 = compute(8):**
  - **Since 8 < 10, the function returns 8 * 2 = 16.**
- **result2 = compute(12):**
  - **Since 12 >= 10, the function returns 12 + 5 = 17.**

Display Result:

- **DISPLAY result1 + result2:**
  - **This displays the sum of result1 and result2.**

result = 16+17 = 33