



MISSION: BINARY SEARCH PROTOCOL

STATUS: ACTIVE

THE SCENARIO

You are a Backend Engineer. You have a sorted database of User IDs.
A linear search takes too long. You must implement a "Binary Search" algorithm to find a specific ID and report the efficiency.



OBJECTIVE

Find the target ID: 23

From the list: [2, 5, 8, 12, 16, 23, 38, 56, 72, 91]



CONSTRAINTS

1. No .contains() or linear loops (for-each).
2. You must count every step (iteration).
3. If the ID is found, print the Index and Total Steps.



QUESTION 1: WRITE PSEUDO CODE

(Write the logic on paper first using AP CSP standards)

VARIABLES:

list = [...], target = 23

low = 0, high = length - 1, steps = 0, found = false

// WRITE YOUR LOGIC HERE:

SET low \leftarrow 0

SET high \leftarrow LENGTH(list) - 1

SET steps \leftarrow 0

SET found \leftarrow false

REPEAT WHILE low \leq high AND found = false

INCREMENT steps BY 1

SET mid \leftarrow (low + high) DIV 2

IF list[mid] = target THEN

SET found \leftarrow true

DISPLAY "ID found at index: " + mid

DISPLAY "Total steps: " + steps

ELSE IF list[mid] < target THEN

SET low \leftarrow mid + 1

ELSE

SET high \leftarrow mid - 1

END REPEAT

IF found = false THEN

DISPLAY "ID not found"

DISPLAY "Total steps: " + steps

END PROCEDURE

? QUESTION 2: WRITE SWIFT CODE
(Implement the logic below in Swift Playgrounds)

import Foundation

// 1. DATA INPUT

let database = [2, 5, 8, 12, 16, 23, 38, 56, 72, 91]

let targetID = 23

// 2. YOUR CODE GOES HERE...

```
import Foundation

// 1. DATA INPUT
let database = [2, 5, 8, 12, 16, 23, 38, 56, 72, 91]
let targetID = 23

// 2. INITIAL VARIABLES
var low = 0
var high = database.count - 1
var steps = 0
var found = false

// 3. BINARY SEARCH
while low <= high && found == false {

    steps += 1
    let mid = (low + high) / 2

    if database[mid] == targetID {
        found = true
        print("ID found at index: \(mid)")
        print("Total steps: \(steps)")
    }
    else if database[mid] < targetID {
        low = mid + 1
    }
    else {
        high = mid - 1
    }
}

// 4. IF NOT FOUND
if found == false {
    print("ID not found")
    print("Total steps: \(steps)")
}
```

/*

CONSOLE OUTPUT (EXPECTED RESULT):

Your code MUST produce this exact output:

> SYSTEM: Starting Binary Search Protocol...

> Target ID: 23

> -----

> Step 1: Checking Index [4] -> Value: 16

> Step 2: Checking Index [7] -> Value: 56

> Step 3: Checking Index [5] -> Value: 23

>

> SUCCESS: User ID found at Index 5

> EFFICIENCY: Operation completed in 3 steps.

SYSTEM: Starting Binary Search Protocol...

Target ID: 23

Step 1: Checking Index [4] -> Value: 16

Step 2: Checking Index [7] -> Value: 56

Step 3: Checking Index [5] -> Value: 23

SUCCESS: User ID found at Index 5

EFFICIENCY: Operation completed in 3 steps.

None

```
**BINARY SEARCH PROTOCOL**
```

```
**INITIALIZATION:**
```

```
*   **VARIABLES:** `list` = [2, 5, ... 91], `target` = 23
*   `low` = 0
*   `high` = LENGTH(`list`) - 1
*   `found` = false
```

```
**SEARCH LOOP:**
```

```
*   **REPEAT** until (`low` > `high`)
    1. Calculate the middle index: `mid` = FLOOR((`low` +
`high`) / 2) *(Integer Division)*
    2. **IF** (`list`[`mid`] = `target`)
        *   **OUTPUT** ("Found at index " + `mid`)
        *   `found` = true
        *   **EXIT LOOP** *(Break the loop)*
    3. **ELSE**
        *   **IF** (`list`[`mid`] < `target`)
            *   Adjust the lower bound: `low` = `mid` + 1 *(Go to
the Right Half)*
        *   **ELSE**
            *   Adjust the upper bound: `high` = `mid` - 1 *(Go
to the Left Half)*
```