Problem Title: Search in a Sorted List Without Multiplication, Division, or Bit-Shifts

Company: Netflix

Scenario:

You are working on a constrained system where certain operations (multiplication, division, and bit-shifting) are **not allowed** due to hardware limitations. However, you still need to **efficiently search** for a given element `x` in a **sorted list of integers**.

Your task is to determine if the element exists in the list in **O(log N)** time.

Problem Statement:

Given:

- A sorted list of integers `arr` of length `N`.
- A target value `x`.

Return `true` if `x` exists in the list, otherwise return `false`.

You **cannot** use multiplication (`*`), division (`/`), or bit-shift (<<, >>) operations.

Input Format:

- First line: integer `N` (size of the list).
- Second line: `N` integers (sorted in ascending order).
- Third line: integer `x` (target value).

Output Format:

- `true` if `x` is found, otherwise `false`.

Example 1:

```
Input:
N = 7
```

```
arr = [-5, -2, 0, 3, 7, 10, 15]
x = 7

Output:
true
```

Example 2:

```
Input:
N = 5
arr = [1, 2, 4, 8, 16]
x = 3

Output:
false
```

Approach Hints:

- Use **Binary Search** (O(log N)).
- To calculate `mid`, avoid `(low + high) / 2` since division is not allowed.
  - Instead, use **repeated addition/subtraction** or
  - Use `mid = low + ((high - low) >> 1)` conceptually, but implement without bit shifts.

Possible workaround:

```
mid = low + (high - low) // 2
```

…but implement your own function for integer division using subtraction or addition.

Practice Links:

- LeetCode – Binary Search
- GeeksforGeeks – Binary Search without Division

Video Explanations:

- NeetCode – Binary Search Explained
- Tushar Roy – Binary Search