Recursive binary search algorithm

- **Input**:
  - book_list: Sorted array of book titles
  - low: Starting index of the range.
  - high: Ending index of the range.
  - target_book: The book to search for
- **Output**:
  - The index of target_book in book_list, or -1 if target_book is not found.
- **Steps**:
  1. **Base case:**
     a) If low > high, return -1
  2. **Calculate Middle Index:**
     a) Set mid = (low + high) / 2

Recursive binary search algorithm

3. **Compare Middle Element:**
   a) If book_list[mid] == target_book:
      I. return mid.
   b) Else if book_list[mid] < target_book, recursively call the function:
      I. recursiveBinarySearch(book_list, mid + 1, high, target_book).
   c) Else, recursively call the function:
      I. recursiveBinarySearch(book_list, low, mid - 1, target_book).
4. **Output the Result:**
   3. Return the result from the recursive calls.

Example

indexes	Book titles	
0	Advanced Physics	low
1	Anatomy and Physiology	
2	Biology Essentials	
3	Calculus I	
4	Chemistry Basics	
5	Civil Engineering Principles	
6	Computer Networks	
7	Data Structures	
8	Discrete Mathematics	
9	Economics 101	high

Time Complexity of Recursive Binary Search

- **Best Case**: O(1) – The target is found on the first comparison, meaning mid == target_book.
- **Average Case**: O(log n) – The target is somewhere in the middle of the search space
- **Worst Case**: O(log n) – The target is at either end of the search range, and the recursion goes all the way down to the base case without finding the target.

Space Complexity of Recursive Binary Search

- Recursive Binary Search: O(log n) space, as it requires stack space for each recursive call.

- Recursive Nature:
  - Each recursive call adds a new frame to the call stack.
  - These frames store variables such as low, high, mid, and the return address for the function.

Call Stack

- The call stack is a special region of memory that keeps track of function calls in a program.
- Each time a function is called, a new stack frame is created for that function call. This frame contains all the information the function needs to execute, including:
  - Local variables (like low, high, mid).
  - The function's return address (where to return after the function finishes).
  - The execution context for that function call.