Strivers-A2Z-DSA-Sheet-main\02.Binary Search\1D Arrays\08.Find_peak_element.cpp

```cpp
/*
QUESTION:-
A peak element is an element that is strictly greater than its neighbors.

Given a 0-indexed integer array nums, find a peak element, and return its index. If the array contains multiple peaks, return the index to any of the peaks.

You may imagine that nums[-1] = nums[n] = -∞. In other words, an element is always considered to be strictly greater than a neighbor that is outside the array.

You must write an algorithm that runs in O(log n) time.

Example 1:
Input: nums = [1,2,3,1]
Output: 2
Explanation: 3 is a peak element and your function should return the index number 2.

Example 2:
Input: nums = [1,2,1,3,5,6,4]
Output: 5
Explanation: Your function can return either index number 1 where the peak element is 2, or index number 5 where the peak element is 6.
*/

/*
APPROACH:-
We can use the binary search approach to find the peak element.
1. Initialize low = 0 and high = n-1, where n is the size of the array.
2. While low < high, calculate mid = low + (high - low) / 2.
3. If nums[mid] < nums[mid+1], it means a peak element exists on the right side of mid, so update low = mid+1.
4. Otherwise, a peak element exists on the left side of mid or mid itself is a peak, so update high = mid.
5. After the loop ends, low will be pointing to the peak element index.
6. Return low as the result.

CODE:-
*/

int findPeakElement(vector<int>& nums) {
    int low = 0, high = nums.size()-1;
    while(low < high){
        int mid = low + (high - low) / 2;
        if(nums[mid] < nums[mid+1])
            low = mid+1;
        else
            high = mid;
    }
    return low;
}

// TIME COMPLEXITY: O(log n)
```

```
48 // SPACE COMPLEXITY: O(1)
49
```