Find Peak Element C++

Task 03: Find Peak Element (Divide and Conquer)
You are given an array a[] of n integers. A peak element is an element that is greater or equal to its
neighbors. Your task is to find any one peak element using the divide and conquer strategy.
Complete the function:
int findPeakElement(int a[], int left, int right);
Input Format:
• First line: Integer n (1 ≤ n ≤ 105
)
• Second line: n space-separated integers
Output Format:
• Print the value of any one peak element.
Sample Input:
6
1 3 20 4 1 0
Sample Output:
20
Hints:
• A peak is an element greater than or equal to both of its left and right neighbors (if they exist).
• Use binary search:
– Compute the middle index: mid = (left + right) / 2
– If a[mid] is a peak, return it.
– If a[mid - 1] > a[mid], search the left half.
– Otherwise, search the right half.

Solve in cpp and explain clearly

Here's how to solve **Task 03: Find Peak Element using Divide and Conquer (Binary Search)** in C++:

✅ Understanding the Problem

You're given an array `a[]` of `n` integers. A **peak element** is one that is **greater than or equal to its neighbors**. You need to find **any one peak**, not necessarily the largest.

🔍 Divide and Conquer (Binary Search) Strategy

The idea is:

1. Pick the middle element `a[mid]`.
2. If it's a peak, return it.
3. If the **left neighbor** `a[mid - 1]` is **greater**, then a peak must be on the **left side**.
4. If the **right neighbor** `a[mid + 1]` is **greater**, then a peak must be on the **right side**.

This works because **if you're not at a peak**, you can always move to a neighbor with a higher value — eventually you'll reach a peak.

✅ C++ Code

cpp

```cpp
#include <iostream>
using namespace std;

int findPeakElement(int a[], int left, int right) {
    int mid = left + (right - left) / 2;

    // Check if mid is a peak
    bool leftOk = (mid == 0 || a[mid] >= a[mid - 1]);
    bool rightOk = (mid == right || a[mid] >= a[mid + 1]);

    if (leftOk && rightOk) {
        return a[mid]; // a[mid] is a peak
    }

    // If left neighbor is greater, search left
    if (mid > 0 && a[mid - 1] > a[mid]) {
        return findPeakElement(a, left, mid - 1);
    } else { // else search right
        return findPeakElement(a, mid + 1, right);
    }
}
```

```cpp
int main() {
    int n;
    cin >> n;
    int a[n];

    for (int i = 0; i < n; i++) {
        cin >> a[i];
    }

    cout << findPeakElement(a, 0, n - 1) << endl;
    return 0;
}
```

📘 Explanation

- **Base Case**: If the middle element `a[mid]` is a peak, return it.
- **Recursive Case**: Based on neighbors, search left or right half.
- **Time Complexity**: `O(log n)` — Because each step cuts the array in half.

✅ Sample Input

```
6
1 3 20 4 1 0
```

✅ Output

```
20
```

`20` is a peak because it's greater than both its neighbors `3` and `4`.

If you'd like the **iterative version** or want to **return the index** instead of value, I can provide that too.

Do you like this personality?