Description:

C program that builds a sorted 0..n-1 array and uses a recursive binary search (O(log n)) to find a key. Prints the index if found (or “not found”) and measures CPU time with clock().\

Source Code:

```
#include <stdio.h>

#include <time.h>


int recursiveBinarySearch(int arr[], int low, int high, int key) {

    if (low > high) {

    return -1; // Element not found

    }


    int mid = low + (high - low) / 2; // Safer mid calculation


    if (arr[mid] == key) {

    return mid;

    }

    else if (arr[mid] > key) {

    // Search the left half

    return recursiveBinarySearch(arr, low, mid - 1, key);

    }

    else {

    // Search the right half

    return recursiveBinarySearch(arr, mid + 1, high, key);

    }
```

```
}

int main() {
        int n = 100000;
        int arr[n];

        for (int i = 0; i < n; i++) {
        arr[i] = i;
        }

        int key = 99999;

        clock_t start, end;
        double cpu_time_used;

        start = clock();

        // Initial call: low=0, high=n-1
        int result = recursiveBinarySearch(arr, 0, n - 1, key);

        end = clock();
        cpu_time_used = ((double)(end - start)) / CLOCKS_PER_SEC;
        if (result != -1) {
        printf("Element found at index %d\n", result);
        } else {
        printf("Element not found\n");
```

```
        }
return 0;
}
```

Output:

```
Element found at index 99999
Execution time: 0.000002 seconds
```