Check number exists in array in C++

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

int array11(int nums[], int index, int length) {
    if (index >= length) {
        return 0;
    }
    int small = array11(nums, index + 1, length);
    if (nums[index] == 11) {
        return 1 + small;
    } else {
        return small;
    }
}

int main() {
    int arr[] = {1, 11, 3, 11, 11, 11};
    int length = sizeof(arr) / sizeof(arr[0]);
    cout << array11(arr, 0, length) << endl;
    return 0;
}
```

Initial Call:

array11(arr, 0, 6)

- **Condition:** index = 0, length = 6 → index < length is true.
- **Value at nums[0]:** 1 (not equal to 11).
- **Recursive Call:**

  array11(arr, 1, 6)

Second Call:

array11(arr, 1, 6)

- **Condition:** index = 1, length = 6 → index < length is true.
- **Value at nums[1]:** 11 (equal to 11).
- **Recursive Call:**

  array11(arr, 2, 6)

Third Call:

array11(arr, 2, 6)

- **Condition:** index = 2, length = 6 → index < length is true.
- **Value at nums[2]:** 3 (not equal to 11).
- **Recursive Call:**

  array11(arr, 3, 6)

Fourth Call:

array11(arr, 3, 6)

- **Condition:** index = 3, length = 6 → index < length is true.
- **Value at nums[3]:** 11 (equal to 11).
- **Recursive Call:**

  array11(arr, 4, 6)

Fifth Call:

array11(arr, 4, 6)

- **Condition:** index = 4, length = 6 → index < length is true.
- **Value at nums[4]:** 11 (equal to 11).
- **Recursive Call:**

	array11(arr, 5, 6) **Sixth Call:** array11(arr, 5, 6) • **Condition:** index = 5, length = 6 → index < length is true. • **Value at nums[5]:** 11 (equal to 11). • **Recursive Call:** array11(arr, 6, 6) **Base Case (Seventh Call):** array11(arr, 6, 6) • **Condition:** index = 6, length = 6 → index >= length is true. • **Action:** Return 0. **Backtracking and Return Values:** 1. **Sixth Call:** o **Value at nums[5]:** 11 → Return 1 + 0 = 1. 2. **Fifth Call:** o **Value at nums[4]:** 11 → Return 1 + 1 = 2. 3. **Fourth Call:** o **Value at nums[3]:** 11 → Return 1 + 2 = 3. 4. **Third Call:** o **Value at nums[2]:** 3 → Return 0 + 3 = 3. 5. **Second Call:** o **Value at nums[1]:** 11 → Return 1 + 3 = 4. 6. **Initial Call:** o **Value at nums[0]:** 1 → Return 0 + 4 = 4.
Output:- 4	