Trạng thái	Đã xong
Bắt đầu vào lúc	Thứ Ba, 30 tháng 1 2024, 11:26 PM
Kết thúc lúc	Thứ Tư, 31 tháng 1 2024, 2:12 PM
Thời gian thực hiện	14 giờ 46 phút
Điểm	6,00/7,00
Điểm	**8,57** trên 10,00 (**85,71**%)

Câu hỏi 1
Đúng
Đạt điểm 1,00 trên 1,00

The prices of all cars of a car shop have been saved as an array called N. Each element of the array N is the price of each car in shop. A person, with the amount of money k want to buy as much cars as possible.

Request: Implement function

buyCar(int* nums, int length, int k);

Where `nums` is the array N, `length` is the size of this array and `k` is the amount of money the person has. Find the maximum cars this person can buy with his money, and return that number.

Example:

`nums=[90, 30, 20, 40, 50]; k=90;`

The result is 3, he can buy the cars having index 1, 2, 3 (first index is 0).

Note: The library `iostream`, `'algorithm'` and `using namespace std` have been used. You can add other functions but you are not allowed to add other libraries.

For example:

Test	Result

`int nums[] = {90,30,40,90,20};`
`int length = sizeof(nums)/sizeof(nums[0]);`
`cout << buyCar(nums, length, 90) << "\n";`
	3

Answer: (penalty regime: 0 %)

Reset answer

```
int buyCar(int* nums, int length, int k) {
    sort(nums, nums + length);
    int cnt = 0;
    int firstmoney = k;
    for (int i = 0; i < length; ++i) {
        if (nums[i] <= firstmoney) {
            firstmoney -= nums[i];
            cnt++;
        } else {
            break;
        }
    }
    return cnt;
}

```

	Test	Expected	Got	

✓	`int nums[] = {90,30,40,90,20};`
	`int length = sizeof(nums)/sizeof(nums[0]);`
	`cout << buyCar(nums, length, 90) << "\n";`
		3	3	✓

Passed all tests! ✓

Đúng

Marks for this submission: 1,00/1,00.

Câu hỏi **2**

Đúng

Đạt điểm 1,00 trên 1,00

Given an array of integers.

Your task is to implement a function with the following prototype:

```
bool consecutiveOnes(vector<int>& nums);
```

The function returns if all the `1`s appear consecutively in `nums`. If `nums` does not contain any elements, please return `true`

Note:

- The `iostream` and `vector` libraries have been included and `namespace std` are being used. No other libraries are allowed.
- You can write helper functions.
- Do not use global variables in your code.

For example:

Test	Result

`vector<int> nums {0, 1, 1, 1, 9, 8};`
`cout << consecutiveOnes(nums);`
	1

Answer: (penalty regime: 0 %)

Reset answer

```
bool consecutiveOnes(vector<int>& nums) {
    // STUDENT ANSWER
    int cnt = 0;
    int pos = 0;
    int size = nums.size();
    for (int i = 0; i < size; ++i)
        {
        if (nums[i] == 1) ++cnt;
        }
    for (int i = 0; i < size; ++i)
    {
    if (nums[i] == 1)
        {
        pos = i;
        break;
        }
    }
    for (int i = pos; i < pos + cnt; ++i)
        {
        if (nums[i] != 1) return false;
        }
    return true;
}
```

	Test	Expected	Got	

✓	vector<int> nums {0, 1, 1, 1, 9, 8};
	cout << consecutiveOnes(nums);
		1	1	✓

✓	vector<int> nums {};
	cout << consecutiveOnes(nums);
		1	1	✓

✓	vector<int> nums {0, 1, 1, 1, 2, 2, 2, 4, 5, 5, 5, 5, 5, 6, 6, 6, 6, 7, 7, 8};
	cout << consecutiveOnes(nums);
		1	1	✓

✓	vector<int> nums {0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9}; cout << consecutiveOnes(nums);	1	1	✓
✓	vector<int> nums {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9}; cout << consecutiveOnes(nums);	1	1	✓

	Test	Expected	Got	
✓	vector<int> nums {3, 0, 8, 8, 2, 9, 0, 4, 8, 4, 0, 9, 5, 0, 5, 9, 6, 2, 5, 4, 5, 1, 6, 6, 1, 0, 2, 6, 8, 4, 7, 7, 2, 5, 4, 7, 4, 1, 4, 3, 5, 5, 6, 5, 8, 6, 1, 7, 8, 4, 6, 6, 1, 2, 2, 5, 0, 6, 3, 6, 8, 2, 8, 6, 1, 1, 8, 6, 7, 7, 4, 6, 9, 2, 5, 0, 2, 9, 8, 9, 5, 0, 9, 8, 0, 7, 3, 3, 1, 8, 2, 2, 9, 5, 5, 6, 3, 0, 2, 5, 5, 3, 7, 2, 7, 4, 8, 4, 2, 4, 5, 2, 0, 0, 6, 4, 6, 4, 9, 9, 7, 3, 9, 1, 9, 4, 4, 0, 8, 4, 1, 4, 0, 0, 9, 6, 5, 0, 4, 4, 6, 3, 1, 9, 5, 2, 0, 8, 7, 9, 6, 7, 5, 8, 3, 9, 3, 7, 2, 0, 6, 1, 0, 9, 6, 0, 5, 3, 0, 6, 6, 9, 4, 2, 7, 0, 4, 5, 9, 6, 8, 3, 9, 0, 5, 1, 0, 8, 1, 5, 9, 1, 5, 2, 4, 4, 2, 7, 9, 4, 6, 6, 3, 3, 8, 6, 8, 2, 1, 5, 8, 4, 0, 5, 9, 5, 5, 2, 2, 3, 1, 8, 6, 3, 1, 2, 2, 3, 2, 4, 4, 1, 4, 4, 8, 6, 4, 1, 2, 6, 6, 5, 5, 2, 5, 3, 2, 6, 4, 5, 2, 3, 9, 6, 0, 8, 8, 9, 1, 7, 0, 3, 4, 8, 4, 1, 7, 9, 2, 9, 4, 6, 3, 5, 9, 8, 6, 1, 8, 2, 7, 2, 1, 5, 3, 0, 6, 8, 0, 1, 6, 1, 1, 6, 0, 6, 5, 8, 9, 3, 2, 1, 3, 3, 6, 1, 7, 9, 5, 9, 0, 2, 0, 6, 9, 1, 9, 0, 7, 4, 6, 4, 3, 2, 3, 5, 1, 4, 1, 6, 1, 9, 0, 8, 8, 4, 4, 6, 6, 4, 0, 2, 6, 6, 6, 9, 2, 9, 6, 7, 9, 2, 8, 5, 3, 4, 7, 3, 8, 7, 3, 2, 8, 1, 9, 8, 3, 5, 1, 2, 1, 0, 7, 2, 7, 1, 1, 3, 1, 7, 0, 7, 3, 6, 0, 7, 1, 7, 2, 1, 2, 7, 1, 2, 7, 7, 3, 1, 4, 8, 3, 7, 9, 9, 6, 1, 0, 3, 7, 6, 4, 4, 9, 6, 1, 5, 6, 3, 0, 4, 0, 7, 5, 0, 1, 0, 1, 9, 1, 1, 9, 4, 4, 4, 2, 9, 7, 2, 2, 7, 2, 5, 6, 4, 5, 9, 3, 4, 6, 4, 7, 8, 6, 9, 0, 2, 9, 4, 3, 3, 6, 6, 8, 6, 4, 0, 3, 7, 3, 0, 0, 0, 0, 0, 0, 5, 9, 0, 2, 3, 6, 9, 5, 6, 4, 5, 7, 3, 3, 2, 7, 1, 3, 2, 2, 7, 1, 6, 4, 8, 6, 7, 9, 4, 3, 1, 5, 8, 8, 9, 3, 1, 0, 9, 3, 8, 3, 4, 6, 7, 3, 7, 2, 9, 9, 1, 9, 4, 5, 3, 9, 0, 1, 3, 4, 6, 7, 7, 0, 9, 7, 0, 7, 3, 5, 1, 9, 0, 9, 9, 8, 5, 9, 2, 0, 9, 2, 9, 4, 7, 1, 4, 5, 4, 7, 5, 8, 8, 8, 7, 0, 3, 1, 8, 7, 5, 6, 6, 8, 6, 2, 6, 6, 4, 4, 0, 8, 3, 5, 4, 8, 8, 1, 4, 3, 9, 2, 5, 5, 5, 5, 3, 6, 7, 0, 4, 5, 5, 9, 6, 0, 2, 8, 7, 4, 5, 2, 1, 0, 2, 7, 6, 4, 4, 2, 0, 0, 9, 4, 1, 4, 2, 6, 7, 8, 1, 7, 6, 9, 6, 9, 1, 8, 4, 5, 2, 2, 0, 9, 3, 8, 1, 4, 4, 9, 4, 3, 3, 5, 5, 7, 7, 8, 4, 5, 6, 5, 2, 8, 6, 3, 1, 6, 0, 8, 2, 9, 2, 1, 0, 9, 5, 2, 5, 3, 7, 2, 6, 8, 9, 2, 0, 1, 0, 6, 1, 4, 4, 9, 3, 1, 7, 8, 3, 8, 9, 2, 8, 8, 7, 9, 9, 6, 4, 7, 8, 4, 7, 0, 7, 1, 0, 0, 0, 5, 3, 5, 1, 5, 1, 4, 5, 0, 9, 8, 7, 5, 4, 2, 6, 1, 9, 5, 8, 6, 3, 7, 8, 3, 2, 5, 4, 0, 4, 0, 6, 9, 0, 6, 1, 8, 3, 9, 8, 1, 2, 5, 7, 3, 2, 3, 3, 2, 1, 7, 2, 1, 8, 4, 8, 2, 3, 6, 5, 5, 0, 7, 6, 7, 9, 4, 2, 9, 5, 9, 0, 2, 5, 9, 1, 3, 1, 1, 4, 9, 3, 1, 7, 7, 9, 8, 9, 2, 0, 1, 5, 5, 5, 4, 7, 3, 4, 7, 1, 5, 6, 2, 2, 3, 2, 9, 8, 3, 7, 8, 6, 8, 8, 2, 8, 7, 8, 2, 0, 5, 7, 3, 4, 0, 7, 4, 4, 1, 8, 8, 4, 0, 6, 6, 5, 5, 3, 1, 7, 8, 9, 5, 9, 7, 9, 5, 5, 9, 5, 5, 9, 4, 4, 0, 7, 5, 0, 4, 9, 1, 3, 2, 2, 3, 9, 8, 2, 2, 9, 0, 6, 1, 4, 6, 9, 0, 9, 4, 9, 8, 2, 0, 8, 1, 0, 8, 1, 4, 8, 9, 5, 1, 1, 0, 6, 2, 7, 0, 5, 5, 1, 6, 0, 8, 2, 0, 3, 7, 2, 1, 1, 9, 7, 2, 3, 7, 2, 1, 0, 4, 1, 4, 7, 3, 7, 9, 2, 0, 5, 3, 8, 5, 6, 8, 0, 3, 3, 2, 0, 2, 8, 9, 3, 6, 3, 5, 1, 6, 8, 8, 1, 1, 0, 9, 1, 5, 4, 6, 0, 4, 6, 6, 3, 4, 0, 7, 8, 8, 5, 8, 3, 1, 7, 8, 2, 9, 3, 9, 1, 4, 4, 3, 3, 0, 6, 9, 1, 6, 6, 4, 1, 2, 6, 0, 0, 6, 1, 2, 1, 3, 6, 0, 4, 1, 8, 9, 3, 9, 7, 1, 0, 0, 0, 6, 8, 3, 5, 3, 3, 8, 7, 0, 8, 5, 7, 2, 9, 8, 9, 2, 8, 9, 7, 7, 5, 0, 6, 9, 8, 9, 9, 3, 1, 7, 1, 5, 2, 9, 5, 4, 1, 8, 4, 5, 9, 3, 9, 1, 9, 5, 0, 4, 9, 7, 7, 3, 6, 8, 8, 7, 8, 1, 8, 2, 4, 5, 3, 5, 3, 8, 3, 7, 5, 3, 9, 7, 3, 2, 3, 2, 5, 9, 5, 1, 9, 7, 8, 7, 9, 7, 8, 2, 3, 2, 4, 3, 3, 7, 6, 1, 0, 1, 9, 5, 7, 8, 0, 9, 3, 5, 5, 3, 2, 5, 2, 3, 3, 0, 0, 2, 2, 1, 1, 8, 8, 4, 3, 3, 8, 3, 4, 2, 7, 0, 7, 3, 3, 8, 9, 0, 4, 0, 3, 5, 6, 1, 9, 1, 5, 0, 4, 5, 3, 0, 3, 0, 0, 7, 4, 1, 1, 5, 5, 7, 2, 9, 0, 7, 3, 1, 5, 3, 4, 3, 2, 7, 2, 5, 0, 9, 3, 1, 2, 7, 4, 8, 2, 2, 7, 7, 7, 0, 9, 1, 4, 4, 2, 0, 4, 2, 6, 0, 3, 3, 7, 2, 8, 4, 0, 5, 0, 9, 6, 7, 6, 1, 9, 8, 1, 9, 2, 6, 8, 7, 9, 7, 2, 7, 8, 5, 0, 7, 5, 0, 1, 3, 3, 3, 8, 7, 1, 7, 2, 2, 1, 8, 5, 0, 1, 0, 0, 3, 2, 4, 2, 8, 1, 5, 8, 5, 8, 1, 8, 9, 9, 9, 3, 4, 8, 5, 0, 7, 4, 9, 8, 1, 9, 3, 5, 5, 3, 6, 3, 5, 3, 0, 5, 0, 9, 5, 8 ...snip... , 3, 5, 8, 2, 3, 6, 0, 6, 8, 8, 8, 8, 0, 6, 4, 5, 9, 0, 0, 2, 0, 5, 4, 9, 5, 7, 4, 9, 1, 6, 1, 4, 3, 6, 7, 2, 4, 9, 1, 3, 0, 5, 3, 0, 7, 2, 3, 7, 2, 2, 7, 4, 5, 9, 0, 2, 9, 0, 9, 7, 7, 5, 8, 7, 0, 6, 6, 3, 6, 3, 0, 3, 6, 4, 2, 8, 2, 8, 6, 9, 5, 6, 9, 4, 7, 2, 2, 4, 1, 8, 2, 7, 8, 7, 0, 3, 6, 6, 8, 2, 3, 2, 5, 9, 7, 3, 8, 7, 9, 3, 0, 2, 4, 2, 5, 9, 3, 9, 1, 8, 9, 1, 8, 7, 7, 8, 4, 3, 9, 7, 6, 0, 2, 6, 8, 7, 2, 1, 6, 4, 0, 1, 7, 5, 8, 0, 7, 2, 9, 0, 2, 4, 1, 2, 6, 2, 5, 4, 7, 9, 6, 5, 1, 7, 3, 4, 2, 1,	0	0	✓

	Test	Expected	Got	
	3, 6, 0, 9, 5, 8, 2, 6, 2, 8, 2, 3, 1, 7, 4, 4, 8, 3, 9, 1, 2, 7, 4, 5, 7, 8, 5, 2, 8, 2, 8, 2, 3, 0, 1, 8, 2, 9, 2, 9, 0, 8, 9, 9, 9, 4, 3, 3, 6, 8, 6, 8, 5, 6, 0, 6, 7, 2, 6, 9, 9, 1, 7, 7, 6, 3, 4, 7, 7, 2, 0, 9, 9, 0, 5, 3, 5, 8, 2, 9, 4, 8, 7, 9, 0, 4, 8, 7, 8, 4, 6, 5, 0, 1, 8, 3, 3, 4, 0, 5, 3, 8, 9, 6, 7, 3, 3, 0, 9, 9, 8, 0, 8, 7, 9, 0, 5, 3, 1, 8, 5, 6, 8, 9, 5, 0, 1, 2, 6, 4, 6, 2, 6, 3, 4, 6, 4, 8, 0, 8, 7, 5, 2, 0, 1, 6, 0, 5, 1, 5, 3, 4, 2, 4, 5, 9, 4, 7, 0, 8, 7, 7, 5, 6, 5, 5, 0, 7, 2, 7, 3, 7, 3, 1, 7, 6, 5, 3, 4, 4, 4, 9, 1, 4, 7, 7, 6, 4, 5, 6, 8, 1, 8, 1, 0, 2, 7, 7, 4, 4, 1, 2, 8, 0, 7, 0, 4, 3, 2, 0, 0, 9, 4, 3, 8, 2, 6, 2, 2, 8, 8, 2, 5, 5, 1, 0, 5, 5, 6, 5, 1, 8, 9, 2, 4, 6, 4, 6, 4, 5, 8, 0, 9, 2, 7, 5, 9, 3, 5, 3, 3, 8, 4, 1, 0, 8, 2, 3, 3, 3, 0, 6, 3, 7, 8, 3, 6, 9, 8, 5, 6, 4, 4, 9, 6, 3, 3, 2, 7, 8, 9, 8, 5, 5, 9, 5, 4, 5, 5, 8, 8, 7, 8, 0, 3, 1, 0, 5, 9, 8, 6, 2, 9, 0, 0, 5, 2, 4, 9, 4, 5, 7, 6, 4, 7, 7, 5, 7, 2, 1, 7, 1, 6, 1, 9, 7, 4, 4, 2, 3, 0, 2, 7, 9, 1, 2, 1, 7, 3, 1, 3, 9, 0, 3, 7, 7, 5, 5, 2, 7, 5, 1, 8, 9, 0, 2, 7, 0, 5, 1, 2, 7, 8, 1, 9, 4, 7, 0, 0, 7, 3, 6, 4, 4, 0, 4, 4, 3, 6, 6, 4, 6, 6, 7, 0, 4, 5, 9, 7, 7, 5, 0, 7, 3, 0, 4, 6, 1, 6, 2, 5, 5, 7, 6, 8, 3, 6, 1, 8, 6, 1, 6, 5, 4, 2, 6, 3, 1, 1, 6, 9, 8, 3, 1, 8, 2, 4, 6, 1, 5, 7, 5, 4, 2, 1, 2, 4, 1, 3, 5, 7, 5, 5, 2, 3, 3, 7, 8, 1, 1, 6, 8, 5, 2, 8, 1, 9, 2, 1, 9, 6, 5, 9, 0, 5, 1, 4, 7, 0, 2, 0, 2, 3, 3, 1, 5, 5, 2, 4, 4, 0, 1, 5, 2, 7, 1, 8, 2, 2, 2, 2, 8, 0, 7, 3, 1, 0, 2, 1, 9, 8, 4, 4, 9, 7, 2, 0, 5, 9, 7, 4, 1, 6, 5, 2, 1, 3, 3, 6, 1, 0, 2, 6, 0, 9, 7, 2, 5, 2, 1, 1, 6, 3, 8, 0, 0, 5, 4, 3, 1, 9, 6, 2, 7, 7, 7, 8, 5, 7, 3, 8, 3, 7, 2, 8, 1, 2, 1, 4, 2, 2, 6, 5, 7, 9, 6, 1, 6, 0, 3, 0, 9, 5, 3, 5, 1, 6, 1, 9, 4, 8, 6, 0, 0, 0, 1, 7, 7, 1, 8, 4, 3, 0, 3, 1, 9, 1, 0, 5, 6, 2, 8, 8, 0, 1, 9, 4, 9, 9, 7, 3, 5, 6, 0, 1, 5, 7, 1, 6, 9, 8, 6, 7, 3, 3, 0, 0, 6, 9, 7, 9, 9, 0, 7, 8, 9, 5, 1, 0, 6, 5, 7, 2, 1, 8, 9, 8, 3, 9, 4, 4, 0, 7, 3, 2, 0, 7, 9, 5, 5, 0, 4, 9, 5, 6, 0, 5, 4, 1, 5, 7, 3, 5, 9, 2, 8, 3, 5, 8, 3, 6, 9, 2, 7, 5, 6, 6, 7, 4, 6, 5, 5, 4, 1, 2, 2, 6, 1, 6, 0, 1, 3, 4, 8, 7, 5, 4, 3, 1, 2, 4, 5, 2, 8, 6, 4, 4, 4, 8, 5, 6, 1, 2, 6, 7, 2, 4, 8, 0, 8, 4, 3, 4, 3, 5, 0, 7, 9, 3, 5, 0, 8, 6, 7, 9, 3, 3, 7, 9, 9, 1, 0, 7, 4, 6, 5, 3, 7, 6, 1, 0, 0, 4, 8, 2, 2, 7, 6, 6, 2, 0, 0, 4, 1, 1, 4, 8, 7, 0, 8, 5, 7, 0, 3, 9, 2, 5, 7, 4, 2, 3, 7, 5, 6, 9, 4, 6, 3, 2, 6, 3, 5, 5, 3, 5, 0, 6, 5, 9, 1, 2, 5, 8, 9, 8, 3, 5, 8, 5, 4, 9, 0, 7, 1, 9, 9, 4, 7, 7, 2, 6, 3, 2, 3, 7, 3, 2, 4, 7, 5, 7, 7, 4, 4, 0, 3, 9, 0, 5, 0, 5, 7, 8, 4, 7, 4, 5, 5, 7, 8, 7, 3, 9, 3, 6, 6, 5, 0, 9, 0, 2, 8, 1, 3, 7, 3, 5, 3, 2, 7, 6, 0, 8, 3, 8, 8, 7, 7, 5, 0, 9, 6, 6, 4, 2, 5, 3, 0, 6, 2, 6, 0, 4, 2, 3, 4, 6, 4, 9, 7, 2, 4, 7, 7, 2, 0, 5, 6, 2, 4, 2, 0, 9, 5, 3, 6, 5, 2, 7, 6, 9, 4, 0, 1, 8, 1, 6, 2, 1, 7, 0, 6, 4, 8, 8, 7, 6, 0, 0, 4, 4, 3, 6, 0, 8, 6, 7, 1, 8, 8, 8, 4, 6, 9, 9, 5, 6, 7, 9, 7, 1, 0, 0, 3, 1, 2, 7, 6, 6, 6, 9, 7, 6, 7, 1, 9, 1, 2, 6, 9, 1, 0, 6, 0, 6, 8, 1, 0, 8, 6, 3, 5, 0, 9, 0, 8, 6, 6, 9, 2, 4, 7, 8, 0, 9, 5, 8, 1, 8, 3, 1, 1, 9, 9, 3, 3, 7, 8, 4, 9, 9, 0, 1, 7, 2, 2, 0, 3, 2, 3, 1, 0, 0, 2, 4, 9, 6, 6, 9, 8, 8, 9, 8, 3, 8, 7, 2, 6, 0, 3, 1, 0, 5, 9, 1, 0, 8, 4, 6, 0, 1, 4, 5, 7, 3, 2, 9, 0, 4, 9, 3, 2, 3, 3, 7, 4, 8, 0, 9, 7, 9, 1, 2, 7, 9, 1, 6, 1, 3, 2, 2, 2, 8, 7, 6, 5, 5, 3, 2, 7, 3, 4, 6, 4, 0, 0, 4, 6, 9, 8, 9, 0, 1, 5, 7, 2, 6, 3, 6, 5, 5, 9, 8, 1, 0, 4, 2, 8, 1, 5, 7, 9, 6, 7, 9, 1, 9, 3, 1, 5, 4, 9}; cout << consecutiveOnes(nums);			

	Test	Expected	Got	
✓	vector<int> nums {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 ...snip... , 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,	1	1	✓

	Test	Expected	Got	
	8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9}; cout << consecutiveOnes(nums);			

	Test	Expected	Got	
✓	vector<int> nums {0, 7, 4, 0, 0, 7, 0, 6, 8, 8, 6, 6, 7, 4, 4, 4, 7, 9, 1, 3, 3, 5, 4, 3, 6, 3, 4, 6, 9, 6, 7, 1, 7, 5, 5, 4, 1, 9, 6, 0, 8, 1, 2, 2, 7, 7, 8, 3, 1, 4, 2, 2, 1, 3, 6, 3, 4, 4, 5, 6, 5, 0, 3, 2, 8, 0, 9, 9, 8, 2, 6, 1, 0, 6, 6, 2, 3, 2, 9, 1, 0, 7, 8, 1, 1, 3, 9, 4, 1, 0, 6, 8, 2, 2, 5, 0, 4, 5, 3, 9, 3, 1, 8, 4, 1, 4, 8, 2, 2, 8, 5, 3, 9, 5, 3, 3, 8, 8, 3, 7, 4, 7, 7, 0, 7, 0, 6, 6, 5, 0, 7, 3, 3, 6, 0, 2, 5, 2, 0, 3, 1, 0, 5, 4, 3, 0, 9, 8, 1, 6, 0, 3, 5, 1, 8, 6, 8, 9, 3, 5, 8, 0, 1, 4, 7, 1, 0, 3, 5, 3, 0, 7, 2, 2, 2, 4, 2, 1, 2, 3, 2, 3, 0, 1, 9, 4, 0, 3, 5, 5, 4, 6, 7, 9, 3, 6, 5, 0, 4, 7, 2, 9, 9, 1, 4, 1, 0, 8, 5, 4, 6, 9, 3, 1, 4, 0, 1, 9, 3, 5, 7, 6, 7, 3, 9, 8, 4, 2, 2, 7, 6, 7, 4, 7, 0, 6, 8, 2, 5, 4, 6, 3, 7, 1, 5, 8, 1, 7, 3, 4, 9, 5, 9, 4, 3, 3, 9, 7, 6, 2, 5, 0, 2, 1, 0, 0, 3, 0, 8, 0, 2, 4, 3, 4, 9, 8, 7, 8, 9, 9, 6, 6, 9, 3, 0, 9, 0, 6, 9, 7, 5, 9, 2, 7, 2, 8, 9, 8, 2, 7, 1, 2, 1, 0, 8, 7, 0, 3, 8, 7, 6, 1, 7, 9, 7, 5, 5, 8, 2, 9, 2, 8, 6, 3, 6, 9, 8, 9, 7, 8, 5, 6, 6, 6, 8, 6, 2, 1, 3, 1, 6, 0, 6, 9, 3, 6, 2, 9, 8, 1, 3, 2, 6, 1, 9, 3, 4, 4, 8, 5, 4, 4, 9, 1, 3, 7, 4, 7, 9, 6, 5, 8, 5, 1, 0, 4, 4, 1, 1, 5, 9, 7, 6, 8, 0, 4, 3, 6, 2, 1, 1, 5, 5, 6, 4, 5, 3, 3, 1, 9, 7, 5, 6, 3, 7, 3, 4, 4, 6, 6, 3, 2, 9, 5, 2, 4, 7, 0, 4, 9, 3, 8, 2, 5, 5, 8, 4, 3, 6, 0, 4, 9, 1, 3, 8, 0, 8, 7, 0, 5, 6, 6, 4, 7, 3, 1, 5, 3, 9, 1, 0, 1, 7, 8, 1, 6, 1, 7, 4, 8, 3, 4, 7, 4, 0, 6, 0, 0, 0, 1, 1, 2, 3, 1, 6, 7, 7, 1, 1, 8, 5, 1, 6, 3, 7, 3, 8, 2, 9, 9, 3, 9, 5, 9, 2, 8, 2, 2, 2, 3, 9, 1, 2, 4, 0, 6, 3, 0, 2, 5, 6, 1, 8, 4, 4, 4, 6, 8, 4, 8, 3, 1, 1, 7, 0, 7, 6, 5, 0, 9, 0, 6, 5, 4, 1, 9, 1, 1, 0, 5, 1, 9, 9, 7, 8, 2, 5, 6, 8, 7, 5, 0, 5, 0, 8, 3, 7, 6, 3, 1, 3, 7, 1, 0, 0, 3, 6, 5, 2, 3, 0, 5, 8, 6, 6, 1, 4, 9, 7, 8, 0, 1, 9, 3, 3, 8, 9, 5, 6, 9, 7, 6, 5, 1, 5, 9, 5, 8, 8, 3, 1, 2, 0, 3, 1, 6, 7, 8, 3, 9, 0, 7, 8, 4, 3, 2, 5, 5, 2, 8, 8, 8, 1, 3, 7, 0, 0, 0, 3, 0, 7, 5, 2, 0, 4, 2, 3, 6, 5, 1, 7, 1, 9, 8, 5, 3, 7, 1, 7, 3, 1, 2, 2, 6, 6, 2, 8, 4, 5, 7, 3, 9, 5, 8, 9, 1, 7, 2, 7, 8, 1, 1, 1, 1, 8, 6, 0, 6, 5, 1, 1, 2, 0, 1, 4, 4, 7, 4, 2, 8, 2, 7, 5, 7, 4, 7, 8, 9, 5, 2, 2, 5, 5, 3, 6, 1, 3, 6, 3, 7, 2, 2, 5, 7, 4, 0, 5, 8, 8, 3, 3, 0, 6, 4, 8, 2, 9, 5, 1, 9, 4, 6, 5, 9, 7, 6, 4, 3, 5, 9, 1, 2, 0, 7, 5, 7, 4, 4, 4, 6, 3, 3, 6, 0, 1, 5, 2, 6, 1, 3, 4, 1, 1, 7, 8, 5, 1, 1, 5, 9, 4, 9, 9, 7, 4, 6, 7, 4, 2, 5, 5, 9, 6, 7, 8, 2, 7, 7, 1, 9, 4, 4, 4, 3, 2, 5, 7, 1, 0, 4, 4, 5, 4, 3, 7, 2, 0, 6, 3, 3, 6, 3, 5, 2, 7, 2, 6, 4, 3, 5, 1, 1, 4, 9, 2, 4, 5, 5, 1, 6, 3, 8, 8, 7, 8, 6, 6, 7, 0, 6, 2, 6, 6, 1, 3, 7, 9, 4, 4, 6, 1, 4, 1, 1, 3, 8, 1, 2, 6, 8, 9, 8, 3, 4, 8, 1, 5, 1, 5, 3, 3, 1, 5, 9, 5, 9, 1, 1, 6, 5, 8, 1, 0, 2, 1, 3, 6, 8, 5, 6, 2, 9, 8, 0, 5, 1, 0, 0, 3, 2, 8, 4, 1, 8, 7, 8, 5, 8, 4, 0, 0, 8, 8, 9, 6, 2, 9, 9, 2, 0, 8, 7, 4, 1, 0, 4, 7, 6, 0, 2, 4, 5, 9, 2, 4, 2, 8, 7, 0, 1, 2, 6, 5, 7, 3, 8, 7, 4, 8, 7, 8, 7, 7, 7, 8, 0, 9, 6, 3, 0, 0, 2, 8, 4, 9, 5, 3, 5, 4, 0, 1, 1, 2, 5, 4, 1, 5, 8, 8, 4, 3, 1, 5, 8, 9, 7, 4, 3, 3, 0, 7, 6, 0, 3, 3, 5, 7, 3, 9, 4, 2, 4, 7, 1, 1, 1, 3, 3, 2, 2, 1, 7, 2, 1, 0, 6, 1, 7, 8, 6, 8, 6, 1, 2, 8, 4, 5, 5, 0, 2, 7, 7, 7, 9, 4, 8, 3, 8, 4, 0, 7, 5, 8, 6, 6, 9, 6, 8, 1, 0, 1, 5, 4, 0, 0, 1, 0, 4, 4, 2, 5, 4, 2, 8, 1, 4, 6, 5, 9, 0, 4, 3, 6, 4, 1, 2, 8, 4, 8, 4, 5, 6, 4, 1, 0, 8, 1, 9, 8, 1, 6, 5, 0, 2, 5, 6, 1, 4, 8, 5, 9, 3, 4, 7, 0, 2, 7, 2, 4, 1, 9, 7, 1, 7, 5, 9, 9, 9, 9, 3, 8, 0, 1, 9, 9, 0, 3, 0, 6, 5, 4, 4, 0, 0, 4, 9, 4, 1, 5, 2, 4, 4, 9, 7, 9, 1, 4, 3, 6, 9, 8, 5, 5, 4, 7, 1, 7, 3, 1, 7, 2, 1, 2, 1, 7, 5, 6, 5, 8, 0, 3, 6, 9, 0, 0, 9, 9, 4, 3, 9, 3, 0, 0, 9, 8, 9, 5, 6, 5, 0, 2, 0, 0, 2, 4, 7, 3, 2, 9, 0, 7, 1, 5, 2, 6, 5, 9, 9, 1, 0, 1, 2, 9, 4, 0, 3, 2, 8, 4, 5, 9, 9, 2, 9, 9, 8, 7, 1, 8, 5, 7, 6, 8, 0, 4, 7, 2, 1, 2, 5, 5, 5, 0, 0, 0, 5, 9, 2, 9, 7, 2, 5, 8, 8, 1, 1, 1, 0, 8, 2, 2, 6, 8, 9, 0, 7, 8, 9, 6, 0, 7, 5, 5, 6, 0, 0, 5, 7, 6, 2, 9, 6, 1, 4, 9, 1, 7, 3, 1, 8, 5, 1, 0, 1, 8, 8, 3, 2, 9, 6, 2, 2, 1, 8, 2, 5, 6, 2, 1, 8, 1, 5, 3, 3, 2, 5, 6, 2, 1, 8, 3, 2, 0, 1, 5, 0 ...snip... , 6, 9, 1, 6, 5, 1, 0, 4, 2, 0, 5, 1, 2, 1, 6, 7, 5, 8, 5, 4, 4, 6, 8, 3, 5, 3, 1, 4, 0, 7, 3, 2, 2, 8, 4, 1, 8, 2, 2, 8, 5, 7, 3, 7, 8, 0, 6, 4, 9, 8, 5, 6, 8, 1, 6, 4, 2, 4, 1, 6, 8, 4, 3, 8, 2, 7, 3, 5, 0, 1, 9, 1, 1, 5, 7, 6, 7, 0, 6, 3, 2, 0, 9, 1, 0, 9, 2, 7, 5, 0, 2, 8, 4, 1, 8, 6, 5, 0, 0, 9, 3, 0, 9, 0, 9, 0, 7, 5, 6, 9, 6, 4, 7, 4, 5, 1, 1, 8, 0, 8, 0, 3, 5, 1, 0, 3, 2, 5, 5, 2, 1, 3, 1, 1, 3, 2, 2, 8, 7, 0, 3, 7, 2, 2, 3, 7, 2, 5, 7, 7, 2, 0, 3, 7, 3, 2, 5, 1, 6, 6, 9, 1, 7, 9, 2, 9, 5, 7,	0	0	✓

<table>
<tr><th></th><th>Test</th><th>Expected</th><th>Got</th><th></th></tr>
<tr><td></td><td><pre>
8, 3, 1, 2, 3, 8, 9, 4, 6, 1, 0, 3, 6, 4, 4, 8, 0, 8, 3, 0, 6, 5, 8, 7, 2, 2, 5, 4, 7, 7, 5, 6,
8, 9, 8, 4, 9, 0, 8, 3, 3, 4, 7, 6, 4, 2, 2, 2, 7, 4, 2, 0, 3, 2, 2, 5, 6, 2, 9, 9, 9, 5, 8, 3,
8, 3, 5, 8, 0, 2, 3, 8, 1, 4, 4, 8, 0, 3, 9, 5, 8, 4, 1, 1, 1, 4, 3, 4, 2, 5, 5, 8, 5, 4, 6, 8,
3, 8, 7, 4, 4, 2, 6, 8, 6, 1, 3, 8, 8, 4, 2, 7, 7, 4, 8, 7, 8, 1, 9, 6, 1, 9, 9, 7, 1, 0, 1, 6,
2, 8, 5, 1, 5, 6, 2, 8, 1, 7, 1, 8, 2, 9, 8, 1, 7, 2, 3, 3, 5, 4, 1, 6, 9, 7, 4, 5, 5, 9, 8, 4,
9, 6, 6, 2, 3, 6, 1, 6, 5, 6, 4, 0, 1, 8, 1, 9, 3, 6, 4, 9, 1, 0, 2, 5, 1, 6, 0, 2, 4, 1, 3, 0,
1, 2, 5, 9, 2, 6, 3, 7, 8, 9, 6, 0, 6, 7, 9, 6, 6, 4, 2, 3, 4, 6, 3, 3, 8, 2, 9, 5, 1, 4, 6, 5,
1, 2, 5, 2, 8, 6, 6, 7, 5, 6, 6, 6, 9, 2, 9, 3, 4, 2, 8, 1, 3, 5, 4, 0, 0, 8, 0, 4, 3, 4, 4, 5,
5, 8, 3, 8, 3, 2, 7, 8, 3, 0, 0, 7, 8, 2, 9, 1, 7, 8, 2, 5, 6, 3, 2, 7, 7, 2, 1, 1, 7, 1, 3, 3,
4, 7, 7, 1, 5, 7, 4, 3, 0, 7, 5, 7, 4, 2, 3, 9, 0, 2, 6, 8, 0, 4, 2, 8, 4, 8, 7, 5, 3, 2, 4, 3,
5, 7, 9, 2, 6, 9, 9, 3, 8, 2, 9, 3, 9, 0, 2, 9, 4, 6, 0, 3, 0, 7, 6, 8, 4, 8, 7, 1, 5, 1, 7, 2,
8, 9, 6, 8, 7, 1, 4, 2, 4, 2, 8, 7, 5, 9, 4, 6, 1, 1, 5, 1, 5, 0, 0, 3, 8, 0, 0, 1, 0, 0, 4, 3,
4, 2, 8, 0, 5, 1, 3, 6, 5, 7, 3, 4, 4, 1, 6, 6, 0, 1, 7, 8, 9, 1, 1, 5, 7, 0, 0, 7, 7, 1, 4, 4,
7, 0, 9, 9, 9, 4, 9, 8, 0, 6, 2, 4, 3, 6, 8, 0, 6, 3, 2, 1, 6, 8, 1, 7, 7, 0, 9, 4, 8, 6, 3, 2,
0, 4, 8, 7, 9, 6, 5, 0, 5, 1, 1, 3, 7, 6, 8, 8, 5, 6, 9, 3, 6, 9, 6, 4, 2, 5, 7, 0, 0, 7, 7, 7,
6, 7, 3, 9, 1, 0, 2, 7, 4, 6, 1, 5, 2, 3, 7, 9, 6, 7, 5, 9, 7, 2, 9, 0, 7, 1, 3, 6, 6, 8, 1, 4,
5, 0, 2, 7, 0, 2, 6, 7, 9, 8, 0, 5, 9, 4, 3, 4, 2, 7, 8, 2, 3, 3, 8, 9, 1, 0, 2, 1, 2, 8, 6, 9,
8, 2, 0, 8, 9, 9, 3, 2, 6, 1, 0, 7, 7, 3, 8, 7, 9, 8, 8, 3, 1, 3, 3, 4, 0, 6, 9, 2, 6, 8, 8, 5,
0, 7, 0, 0, 1, 3, 2, 0, 9, 7, 0, 4, 4, 1, 9, 5, 2, 5, 1, 5, 2, 6, 3, 3, 5, 9, 0, 8, 9, 2, 7, 7,
5, 4, 8, 6, 8, 1, 1, 1, 8, 0, 0, 5, 3, 9, 0, 4, 6, 2, 1, 6, 7, 8, 1, 1, 9, 3, 8, 5, 7, 9, 7, 8,
5, 9, 6, 1, 0, 9, 8, 2, 5, 8, 9, 4, 0, 3, 9, 1, 9, 6, 0, 5, 8, 9, 7, 3, 0, 0, 5, 4, 4, 3, 2, 9,
3, 6, 2, 1, 1, 5, 2, 6, 4, 0, 1, 9, 4, 1, 1, 8, 3, 5, 6, 3, 5, 9, 3, 0, 9, 3, 9, 5, 2, 4, 5, 9,
0, 9, 5, 5, 1, 1, 3, 7, 3, 3, 4, 0, 2, 5, 3, 6, 3, 2, 1, 7, 9, 6, 9, 9, 7, 1, 3, 9, 0, 2, 9, 1,
1, 2, 7, 9, 8, 7, 6, 8, 6, 9, 3, 3, 4, 1, 1, 0, 4, 4, 3, 0, 3, 6, 9, 9, 4, 2, 7, 0, 8, 0, 2, 3,
1, 9, 8, 0, 3, 0, 7, 3, 7, 1, 6, 7, 1, 8, 5, 7, 9, 7, 6, 1, 2, 9, 3, 9, 6, 9, 5, 5, 9, 8, 8, 0,
8, 6, 7, 2, 9, 1, 2, 8, 1, 5, 8, 5, 7, 3, 9, 0, 7, 4, 7, 1, 6, 7, 7, 3, 4, 2, 1, 9, 8, 3, 1, 3,
4, 3, 8, 2, 9, 8, 2, 8, 7, 5, 3, 7, 9, 0, 3, 9, 6, 5, 3, 7, 6, 3, 2, 6, 5, 7, 2, 9, 6, 1, 4, 5,
2, 7, 0, 9, 1, 2, 7, 4, 1, 7, 9, 2, 8, 8, 3, 9, 2, 3, 8, 3, 7, 5, 1, 7, 8, 0, 7, 6, 1, 3, 8, 5,
1, 0, 7, 6, 0, 9, 9, 4, 8, 3, 3, 8, 6, 3, 1, 0, 0, 0, 0, 4, 7, 7, 5, 3, 4, 7, 4, 6, 1, 5, 5, 2,
9, 3, 9, 6, 5, 1, 9, 7, 3, 2, 5, 3, 0, 9, 2, 3, 9, 3, 8, 8, 3, 9, 4, 9, 7, 7, 1, 8, 0, 9, 5, 3,
3, 5, 1, 0, 4, 0, 8, 3, 6, 4, 3, 1, 9, 4, 9, 2, 7, 3, 8, 1, 3, 7, 6, 4, 5, 2, 0, 2, 0, 9, 2, 8,
2, 6, 3, 3, 7, 4, 6, 8, 6, 8, 8, 0, 9, 6, 7, 2, 7, 4, 8, 2, 7, 1, 0, 7, 9, 1, 4, 2, 3, 7, 5, 2,
2, 0, 8, 9, 7, 5, 9, 7, 9, 4, 4, 3, 3, 5, 0, 4, 6, 0, 5, 0, 8, 3, 7, 3, 2, 1, 8, 3, 4, 2, 7, 9,
6, 8, 2, 1, 7, 4, 7, 7, 0, 1, 9, 6, 0, 6, 3, 6, 2, 2, 9, 3, 8, 0, 6, 6, 8, 2, 8, 6, 2, 0, 3, 7,
7, 7, 7, 0, 5, 5, 6, 1, 0, 7, 4, 8, 4, 5, 2, 1, 6, 4, 0, 6, 7, 0, 8, 5, 8, 8, 0, 7, 0, 9, 2, 7,
3, 3, 6, 5, 4, 5, 3, 4, 1, 4, 7, 2, 6, 3, 3, 7, 0, 1, 5, 1, 5, 5, 2, 2, 3, 9, 6, 0, 8, 0, 1, 2};
cout << consecutiveOnes(nums);
</pre></td><td></td><td></td><td></td></tr>
</table>

	Test	Expected	Got	
✓	vector<int> nums {7, 5, 9, 7, 3, 5, 6, 0, 1, 1, 8, 0, 3, 9, 9, 7, 9, 4, 7, 1, 9, 5, 5, 5, 3, 5, 4, 5, 7, 5, 4, 5, 7, 8, 7, 6, 1, 6, 9, 4, 1, 4, 0, 9, 5, 2, 1, 8, 9, 6, 6, 8, 6, 0, 3, 3, 3, 3, 1, 9, 2, 1, 3, 0, 5, 0, 0, 6, 7, 0, 6, 3, 8, 4, 0, 0, 7, 2, 4, 5, 7, 3, 9, 1, 2, 4, 5, 8, 9, 4, 4, 7, 9, 0, 0, 7, 5, 7, 4, 4, 7, 9, 3, 2, 1, 2, 2, 7, 2, 1, 3, 5, 9, 3, 9, 8, 5, 7, 5, 8, 0, 5, 7, 8, 7, 2, 3, 2, 1, 1, 4, 8, 1, 0, 3, 7, 6, 0, 7, 9, 1, 2, 3, 1, 6, 9, 5, 1, 5, 3, 2, 3, 6, 6, 2, 6, 4, 5, 1, 4, 1, 3, 6, 3, 6, 6, 9, 7, 1, 3, 8, 3, 8, 8, 1, 1, 2, 1, 8, 3, 0, 2, 6, 0, 2, 0, 6, 8, 0, 3, 8, 1, 0, 4, 4, 0, 6, 7, 0, 5, 0, 9, 5, 1, 8, 1, 1, 3, 4, 8, 6, 1, 7, 9, 4, 2, 0, 8, 2, 6, 6, 1, 4, 4, 0, 4, 9, 5, 3, 1, 1, 7, 3, 6, 5, 1, 1, 3, 4, 7, 0, 6, 6, 5, 4, 6, 0, 8, 6, 7, 8, 3, 4, 6, 0, 6, 2, 9, 8, 9, 0, 2, 0, 8, 2, 1, 6, 8, 5, 0, 4, 9, 9, 5, 4, 8, 3, 9, 3, 1, 7, 5, 0, 2, 2, 2, 7, 8, 7, 7, 7, 4, 7, 9, 6, 0, 9, 2, 8, 5, 8, 3, 1, 9, 1, 6, 0, 3, 6, 7, 4, 5, 3, 0, 7, 7, 2, 7, 1, 0, 9, 2, 8, 2, 2, 2, 5, 0, 4, 1, 7, 3, 6, 9, 6, 9, 0, 3, 3, 0, 3, 9, 6, 6, 3, 2, 5, 5, 4, 9, 6, 0, 8, 4, 2, 5, 1, 7, 8, 9, 2, 3, 5, 1, 7, 0, 0, 1, 0, 8, 0, 0, 6, 7, 6, 7, 1, 0, 5, 1, 8, 3, 7, 7, 6, 3, 7, 2, 7, 3, 9, 8, 7, 0, 1, 7, 1, 2, 8, 5, 5, 5, 4, 4, 5, 7, 4, 7, 8, 3, 6, 7, 4, 7, 0, 4, 5, 4, 8, 1, 0, 5, 3, 9, 5, 2, 5, 1, 4, 9, 9, 9, 6, 7, 8, 8, 3, 2, 3, 6, 0, 2, 9, 8, 3, 7, 8, 8, 4, 6, 6, 8, 8, 3, 3, 8, 8, 6, 2, 0, 3, 9, 1, 7, 7, 7, 2, 7, 6, 0, 4, 1, 5, 5, 5, 5, 3, 4, 8, 3, 7, 0, 2, 6, 4, 1, 7, 1, 2, 6, 1, 1, 6, 3, 6, 5, 1, 1, 4, 5, 0, 5, 0, 3, 9, 3, 7, 2, 4, 5, 5, 6, 4, 4, 3, 4, 6, 5, 4, 8, 1, 7, 4, 6, 9, 0, 3, 2, 8, 3, 7, 5, 1, 5, 5, 6, 2, 8, 4, 7, 8, 8, 0, 0, 5, 7, 4, 2, 4, 5, 4, 3, 7, 7, 3, 6, 0, 6, 2, 7, 3, 4, 7, 8, 3, 3, 3, 9, 7, 0, 8, 9, 1, 1, 7, 7, 2, 1, 4, 9, 6, 3, 0, 1, 5, 4, 2, 6, 7, 7, 0, 7, 0, 6, 1, 8, 4, 8, 0, 9, 6, 2, 2, 1, 5, 7, 1, 9, 7, 4, 8, 6, 6, 1, 2, 3, 9, 7, 3, 8, 4, 0, 4, 8, 9, 9, 8, 8, 6, 6, 0, 2, 5, 4, 9, 8, 0, 2, 6, 7, 6, 7, 8, 5, 5, 7, 5, 0, 0, 2, 3, 1, 1, 3, 7, 8, 3, 0, 2, 2, 5, 7, 2, 0, 5, 8, 2, 6, 2, 8, 3, 2, 3, 2, 1, 9, 0, 9, 2, 4, 6, 8, 5, 2, 5, 2, 0, 2, 4, 5, 5, 1, 7, 5, 0, 4, 8, 8, 8, 9, 2, 3, 8, 3, 0, 9, 5, 3, 7, 1, 6, 6, 6, 1, 7, 5, 6, 8, 3, 5, 3, 2, 4, 6, 9, 9, 5, 3, 3, 5, 0, 9, 0, 8, 8, 6, 3, 0, 4, 5, 3, 0, 4, 0, 7, 9, 0, 4, 0, 5, 9, 3, 4, 4, 1, 2, 7, 7, 3, 4, 9, 0, 3, 2, 3, 6, 4, 7, 7, 6, 9, 5, 0, 9, 0, 6, 9, 9, 2, 8, 7, 4, 1, 1, 0, 0, 7, 5, 7, 9, 4, 0, 0, 1, 1, 0, 0, 5, 7, 0, 0, 0, 9, 8, 8, 8, 9, 4, 7, 3, 1, 6, 3, 3, 0, 0, 7, 9, 3, 7, 7, 2, 1, 3, 7, 0, 3, 1, 3, 9, 4, 9, 0, 4, 0, 1, 1, 9, 3, 7, 1, 5, 9, 3, 6, 2, 4, 6, 1, 7, 0, 9, 1, 7, 3, 3, 8, 9, 1, 3, 4, 0, 5, 0, 9, 0, 7, 3, 1, 5, 7, 3, 7, 8, 6, 4, 6, 8, 9, 2, 4, 0, 3, 0, 5, 2, 0, 9, 0, 3, 7, 2, 8, 5, 1, 5, 9, 4, 8, 5, 6, 6, 7, 8, 8, 0, 5, 1, 9, 3, 7, 7, 6, 8, 5, 5, 7, 7, 6, 6, 9, 7, 8, 0, 0, 4, 1, 6, 5, 5, 3, 3, 3, 6, 8, 7, 2, 1, 2, 0, 1, 5, 8, 7, 0, 1, 1, 6, 4, 3, 4, 6, 8, 5, 1, 9, 8, 5, 8, 7, 0, 9, 7, 3, 0, 6, 5, 1, 7, 8, 4, 4, 3, 7, 7, 1, 5, 4, 6, 0, 6, 0, 8, 2, 6, 3, 6, 7, 1, 1, 9, 9, 2, 0, 7, 0, 0, 0, 1, 2, 5, 5, 8, 4, 1, 9, 1, 8, 1, 1, 6, 0, 3, 6, 1, 3, 0, 7, 5, 9, 1, 5, 1, 5, 0, 0, 6, 3, 4, 4, 9, 7, 4, 9, 0, 4, 8, 1, 4, 2, 3, 6, 6, 8, 9, 9, 2, 3, 9, 2, 7, 4, 8, 3, 9, 4, 6, 9, 7, 8, 4, 7, 1, 3, 9, 8, 6, 3, 7, 2, 1, 2, 1, 3, 6, 6, 0, 2, 0, 1, 5, 4, 6, 9, 6, 7, 4, 2, 8, 6, 8, 8, 8, 7, 6, 0, 3, 4, 8, 2, 3, 7, 2, 2, 3, 1, 4, 3, 0, 2, 0, 3, 3, 9, 3, 6, 1, 3, 5, 9, 2, 0, 3, 4, 0, 3, 3, 2, 4, 6, 8, 5, 9, 8, 3, 8, 1, 9, 7, 2, 7, 9, 3, 6, 3, 4, 2, 1, 9, 3, 2, 6, 0, 4, 1, 6, 2, 7, 3, 6, 9, 0, 1, 1, 9, 7, 2, 6, 9, 7, 8, 6, 3, 5, 0, 3, 3, 5, 5, 5, 9, 3, 2, 4, 2, 6, 6, 7, 3, 9, 1, 3, 8, 3, 6, 5, 6, 4, 0, 9, 2, 1, 5, 9, 4, 4, 1, 2, 1, 1, 1, 3, 1, 9, 2, 2, 3, 2, 1, 5, 3, 6, 6, 4, 9, 5, 3, 3, 3, 3, 3, 1, 2, 7, 6, 9, 7, 9, 3, 8, 7, 9, 3, 3, 6, 5, 7, 9, 5, 5, 1, 5, 6, 0, 3, 2, 0, 0, 3, 7, 0, 8, 4, 4, 4, 8, 0, 6, 4, 1, 7, 0, 1, 8, 5, 2, 9, 9, 4, 5, 7, 7, 5, 8, 6, 3, 7, 9, 7, 7, 2, 6, 8, 3, 7, 3, 1, 6, 3, 9, 1, 9, 8, 6, 5, 6, 7, 5, 2, 5, 6, 8, 2, 0, 8, 8, 1, 6, 9, 1, 5, 6, 8, 3, 3, 8 ...snip... , 7, 7, 9, 4, 0, 1, 9, 5, 4, 7, 2, 9, 8, 2, 7, 7, 5, 1, 0, 6, 1, 0, 8, 1, 5, 8, 4, 1, 9, 5, 9, 4, 3, 8, 5, 1, 6, 0, 9, 4, 3, 5, 3, 4, 2, 9, 7, 7, 8, 9, 4, 1, 8, 1, 8, 8, 7, 6, 9, 9, 8, 2, 8, 0, 6, 1, 3, 7, 6, 4, 0, 2, 5, 3, 5, 2, 6, 9, 3, 2, 4, 7, 9, 5, 8, 0, 6, 9, 6, 1, 3, 6, 1, 2, 9, 8, 4, 6, 1, 0, 0, 1, 9, 7, 0, 7, 5, 7, 5, 7, 8, 6, 7, 7, 4, 0, 4, 2, 6, 3, 7, 5, 4, 5, 3, 5, 9, 4, 3, 7, 3, 7, 7, 9, 5, 5, 9, 1, 8, 1, 0, 6, 5, 6, 3, 1, 0, 4, 4, 8, 5, 6, 8, 4, 1, 8, 0, 8, 9, 6, 8, 9, 6, 5, 3, 7, 6, 1,	0	0	✓

	Test	Expected	Got	
	6, 3, 8, 6, 5, 0, 4, 9, 3, 6, 8, 1, 2, 5, 4, 4, 5, 2, 3, 4, 2, 1, 8, 6, 0, 8, 4, 3, 6, 3, 4, 2, 9, 0, 5, 2, 2, 1, 4, 5, 2, 5, 7, 9, 0, 8, 6, 4, 2, 7, 9, 3, 3, 6, 3, 2, 5, 2, 0, 9, 6, 9, 0, 3, 6, 6, 4, 7, 5, 9, 7, 8, 6, 0, 4, 4, 0, 0, 6, 1, 5, 1, 2, 1, 3, 2, 0, 5, 6, 4, 6, 7, 6, 0, 2, 7, 5, 9, 3, 5, 6, 1, 1, 7, 4, 5, 6, 8, 3, 0, 1, 8, 2, 0, 9, 0, 7, 9, 4, 4, 7, 6, 6, 6, 8, 2, 5, 6, 0, 0, 5, 2, 0, 8, 5, 0, 4, 8, 1, 7, 2, 3, 2, 5, 5, 1, 7, 3, 1, 8, 3, 6, 0, 9, 5, 5, 9, 8, 1, 8, 4, 3, 7, 6, 3, 0, 2, 5, 8, 4, 0, 6, 7, 9, 0, 9, 5, 9, 5, 8, 2, 3, 6, 9, 9, 8, 0, 0, 3, 4, 2, 2, 9, 5, 8, 8, 3, 9, 4, 6, 6, 5, 0, 8, 1, 4, 0, 5, 2, 4, 0, 1, 9, 0, 1, 1, 0, 6, 3, 4, 9, 8, 6, 9, 2, 0, 5, 5, 2, 5, 9, 2, 4, 3, 4, 4, 6, 0, 4, 7, 9, 0, 0, 9, 4, 8, 5, 7, 9, 5, 6, 2, 7, 5, 5, 5, 4, 2, 5, 2, 8, 7, 6, 9, 8, 5, 3, 8, 8, 2, 1, 8, 9, 4, 8, 3, 5, 3, 3, 8, 6, 7, 4, 8, 1, 9, 4, 7, 7, 0, 1, 5, 7, 3, 5, 8, 5, 0, 0, 2, 3, 2, 4, 6, 2, 4, 5, 7, 0, 9, 4, 3, 4, 0, 5, 7, 8, 9, 4, 8, 1, 8, 0, 6, 1, 5, 9, 6, 9, 6, 4, 5, 3, 8, 2, 0, 2, 8, 6, 4, 3, 1, 5, 2, 9, 5, 8, 6, 5, 5, 8, 6, 0, 1, 4, 4, 6, 4, 8, 2, 7, 3, 4, 2, 6, 8, 0, 1, 7, 5, 4, 3, 9, 3, 6, 7, 9, 5, 6, 5, 0, 7, 4, 4, 4, 9, 6, 6, 2, 7, 8, 0, 8, 2, 1, 2, 3, 4, 1, 3, 1, 3, 1, 8, 7, 1, 6, 6, 6, 9, 4, 1, 8, 8, 3, 7, 4, 8, 9, 9, 5, 0, 1, 0, 4, 3, 1, 9, 8, 9, 6, 6, 7, 2, 6, 9, 7, 8, 7, 7, 0, 1, 9, 3, 7, 6, 4, 7, 0, 1, 0, 1, 8, 3, 9, 6, 1, 2, 1, 3, 4, 8, 5, 1, 9, 7, 5, 9, 2, 4, 7, 8, 5, 4, 8, 7, 1, 7, 4, 8, 7, 2, 2, 0, 3, 3, 7, 0, 5, 7, 6, 4, 2, 4, 9, 1, 5, 2, 9, 1, 4, 0, 9, 8, 7, 6, 8, 1, 0, 1, 8, 1, 9, 7, 2, 2, 2, 0, 4, 4, 6, 9, 3, 0, 4, 1, 7, 3, 5, 4, 4, 4, 7, 8, 0, 2, 1, 3, 1, 2, 8, 7, 6, 4, 7, 9, 7, 3, 1, 2, 4, 7, 3, 9, 1, 5, 4, 9, 4, 6, 5, 5, 1, 9, 7, 8, 7, 1, 2, 0, 0, 3, 9, 2, 9, 9, 3, 6, 7, 5, 8, 9, 2, 4, 4, 3, 9, 3, 1, 7, 9, 0, 9, 3, 1, 6, 2, 3, 7, 5, 6, 5, 0, 3, 1, 0, 3, 2, 2, 0, 7, 5, 2, 0, 2, 0, 5, 7, 4, 8, 8, 1, 9, 1, 1, 1, 5, 5, 3, 7, 8, 3, 8, 6, 8, 4, 2, 5, 7, 4, 2, 8, 9, 0, 0, 5, 8, 5, 8, 7, 0, 0, 9, 1, 5, 8, 5, 9, 4, 3, 6, 7, 0, 0, 4, 7, 9, 0, 4, 2, 9, 3, 8, 7, 1, 1, 6, 4, 8, 1, 3, 0, 6, 4, 1, 2, 6, 0, 5, 8, 6, 1, 8, 3, 1, 0, 6, 9, 2, 5, 6, 1, 6, 8, 2, 5, 4, 8, 3, 7, 6, 3, 2, 8, 1, 1, 0, 6, 2, 0, 4, 2, 8, 5, 4, 3, 0, 3, 2, 7, 4, 4, 2, 0, 9, 2, 5, 4, 1, 8, 3, 8, 9, 0, 7, 5, 2, 2, 4, 6, 6, 6, 4, 9, 6, 2, 4, 4, 3, 2, 8, 1, 3, 9, 8, 2, 8, 9, 7, 6, 2, 1, 8, 6, 1, 3, 8, 0, 0, 7, 5, 8, 6, 4, 3, 2, 7, 1, 8, 8, 8, 0, 3, 9, 6, 8, 6, 9, 1, 6, 6, 5, 7, 0, 0, 0, 4, 3, 9, 4, 8, 8, 5, 2, 0, 8, 9, 7, 5, 8, 6, 9, 7, 3, 5, 1, 9, 0, 8, 1, 3, 4, 4, 1, 7, 6, 1, 7, 0, 8, 9, 3, 3, 2, 6, 7, 9, 0, 8, 4, 6, 4, 6, 2, 7, 4, 9, 6, 8, 9, 3, 1, 4, 7, 1, 2, 2, 7, 1, 3, 0, 6, 8, 9, 6, 0, 4, 5, 6, 2, 4, 5, 8, 4, 1, 3, 8, 6, 8, 1, 4, 8, 8, 2, 7, 2, 7, 3, 5, 4, 5, 8, 7, 0, 9, 9, 6, 1, 6, 0, 9, 8, 7, 7, 0, 2, 2, 4, 7, 4, 8, 8, 4, 0, 1, 6, 9, 8, 8, 7, 6, 9, 5, 7, 6, 5, 5, 5, 3, 1, 0, 9, 1, 6, 2, 1, 8, 9, 7, 4, 0, 4, 1, 9, 7, 1, 1, 5, 9, 3, 9, 1, 1, 9, 6, 0, 8, 7, 0, 7, 8, 0, 5, 8, 0, 8, 3, 3, 2, 6, 1, 4, 0, 2, 1, 0, 6, 0, 6, 5, 4, 6, 4, 3, 2, 1, 7, 3, 9, 7, 0, 7, 4, 6, 4, 3, 1, 5, 0, 5, 5, 7, 2, 6, 8, 7, 4, 4, 1, 0, 4, 6, 5, 2, 7, 4, 3, 7, 6, 7, 7, 0, 6, 2, 6, 1, 9, 9, 6, 8, 5, 4, 1, 2, 0, 0, 9, 4, 3, 3, 0, 4, 1, 5, 9, 3, 4, 4, 1, 2, 3, 0, 7, 8, 4, 5, 7, 3, 4, 3, 0, 0, 6, 6, 1, 0, 7, 0, 2, 4, 9, 7, 3, 8, 9, 3, 1, 1, 7, 3, 1, 9, 9, 1, 1, 3, 7, 0, 4, 0, 5, 1, 2, 9, 9, 3, 0, 6, 1, 6, 4, 5, 0, 9, 5, 4, 0, 8, 7, 3, 0, 2, 0, 2, 9, 2, 8, 6, 7, 5, 4, 4, 2, 3, 5, 3, 8, 2, 3, 5, 3, 8, 0, 2, 9, 1, 7, 5, 8, 4, 4, 4}; cout << consecutiveOnes(nums);			

Passed all tests! ✓

Đúng

Marks for this submission: 1,00/1,00.

Câu hỏi **3**

Đúng

Đạt điểm 1,00 trên 1,00

Given an array of integers.
Your task is to implement a function with following prototype:

```
int equalSumIndex(vector<int>& nums);
```

The function returns the smallest index `i` such that the sum of the numbers to the left of `i` is equal to the sum of the numbers to the right.
If no such index exists, return `-1`.

Note:
- The `iostream` and `vector` libraries have been included and `namespace std` is being used. No other libraries are allowed.
- You can write helper functions.

For example:

Test	Result

`vector<int> nums {3, 5, 2, 7, 6, 4};`
`cout << equalSumIndex(nums);`
	3

Answer: (penalty regime: 0 %)

Reset answer

```
int equalSumIndex(std::vector<int>& nums) {
    // STUDENT ANSWER
    int totalSum = 0;
    int leftSum = 0;

    for (int num : nums) {
        totalSum += num;
    }
    for (int i = 0; i < nums.size(); i++) {
        totalSum -= nums[i];
        if (totalSum == leftSum) {
            return i;
        }
        leftSum += nums[i];
    }
    return -1;
}
```

	Test	Expected	Got	

✓	`vector<int> nums {3, 5, 2, 7, 6, 4};`
	`cout << equalSumIndex(nums);`
		3	3	✓

✓	`vector<int> nums {3};`
	`cout << equalSumIndex(nums);`
		0	0	✓

Passed all tests! ✓

Đúng

Marks for this submission: 1,00/1,00.

Câu hỏi **4**

Đúng

Đạt điểm 1,00 trên 1,00

Given an array of strings.
Your task is to implement a function with following prototype:

```
int longestSublist(vector<string>& words);
```

The function returns the length of the longest subarray where all words share the same first letter.

Note:
- The `iostream` and `vector` libraries have been included and `namespace std` is being used. No other libraries are allowed.
- You can write helper functions.

For example:

Test	Result

`vector<string> words {"faction", "fight", "and", "are", "attitude"};`
`cout << longestSublist(words);`
	3

Answer: (penalty regime: 0 %)

Reset answer

```
int longestSublist(vector<string>& words) {
    // STUDENT ANSWER
    int maxLength = 0;
    int currentLength = 1;
    for (size_t i = 1; i < words.size(); ++i) {
            if (words[i][0] == words[i - 1][0]) {
            ++currentLength;
        } else {
            currentLength = 1;
        }
        maxLength = max(maxLength, currentLength);
    }
    return maxLength;
}
```

	Test	Expected	Got	

✓	vector<string> words {"faction", "fight", "and", "are", "attitude"};
	cout << longestSublist(words);
		3	3	✓

✓	vector<string> words {};
	cout << longestSublist(words);
		0	0	✓

Passed all tests! ✓

Đúng

Marks for this submission: 1,00/1,00.

Câu hỏi **5**

Đúng

Đạt điểm 1,00 trên 1,00

Implement methods **ensureCapacity, add, size** in template class **ArrayList** representing the array list with type T with the initialized frame. The description of each method is given in the code.

```
template <class T>
class ArrayList {
protected:
    T* data;        // dynamic array to store the list's items
    int capacity;   // size of the dynamic array
    int count;      // number of items stored in the array
public:
    ArrayList(){capacity = 5; count = 0; data = new T[5];}
```

```
    ~ArrayList(){ delete[] data; }
    void    add(T e);
    void    add(int index, T e);
    int     size();
    void    ensureCapacity(int index);
};
```

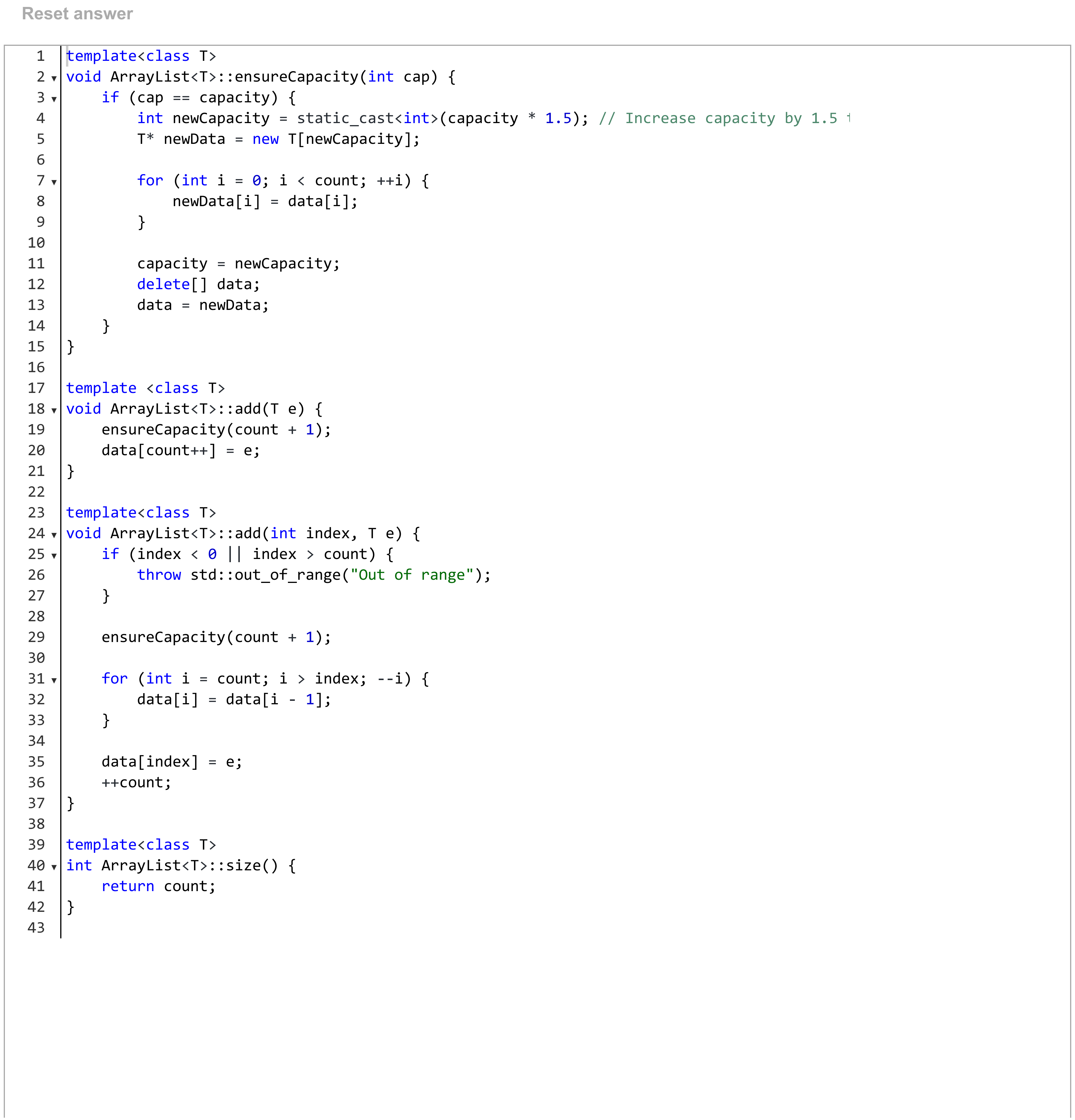
For example:

Test	Result

ArrayList<int> arr;
int size = 10;
 	
for(int index = 0; index < size; index++){
    arr.add(index);
}
 	
cout << arr.toString() << '\n';
cout << arr.size();
	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
	10

ArrayList<int> arr;
int size = 20;
 	
for(int index = 0; index < size; index++){
    arr.add(0, index);
}
 	
cout << arr.toString() << '\n';
cout << arr.size() << '\n';
arr.ensureCapacity(5);
	[19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
	20

Answer: (penalty regime: 0, 0, 0, 0, 0, 100 %)



```cpp
template<class T>
void ArrayList<T>::ensureCapacity(int cap) {
    if (cap == capacity) {
        int newCapacity = static_cast<int>(capacity * 1.5); // Increase capacity by 1.5 t
        T* newData = new T[newCapacity];

        for (int i = 0; i < count; ++i) {
            newData[i] = data[i];
        }

        capacity = newCapacity;
        delete[] data;
        data = newData;
    }
}

template <class T>
void ArrayList<T>::add(T e) {
    ensureCapacity(count + 1);
    data[count++] = e;
}

template<class T>
void ArrayList<T>::add(int index, T e) {
    if (index < 0 || index > count) {
        throw std::out_of_range("Out of range");
    }

    ensureCapacity(count + 1);

    for (int i = count; i > index; --i) {
        data[i] = data[i - 1];
    }

    data[index] = e;
    ++count;
}

template<class T>
int ArrayList<T>::size() {
    return count;
}

```

	Test	Expected	Got	

✓	`ArrayList<int> arr;`
	`int size = 10;`
	
	`for(int index = 0; index < size; index++){`
	`    arr.add(index);`
	`}`
	
	`cout << arr.toString() << '\n';`
	`cout << arr.size();`
		`[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]`
		`10`
			`[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]`
			`10`
				✓

✓	`ArrayList<int> arr;`
	`int size = 20;`
	
	`for(int index = 0; index < size; index++){`
	`    arr.add(0, index);`
	`}`
	
	`cout << arr.toString() << '\n';`
	`cout << arr.size() << '\n';`
	`arr.ensureCapacity(5);`
		`[19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0]`
		`20`
			`[19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0]`
			`20`
				✓

Passed all tests! ✓

Đúng

Marks for this submission: 1,00/1,00.

Câu hỏi **6**

Đúng

Đạt điểm 0,00 trên 1,00

Implement methods **removeAt, removeItem, clear** in template class **ArrayList** representing the singly linked list with type T with the initialized frame. The description of each method is given in the code.

```
template <class T>
class ArrayList {
```

protected:
T* data; // dynamic array to store the list's items
int capacity; // size of the dynamic array
int count; // number of items stored in the array

```
public:
    ArrayList(){capacity = 5; count = 0; data = new T[5];}
   ~ArrayList(){ delete[] data; }
```

```
    void    add(T e);
    void    add(int index, T e);
    int     size();
    bool    empty();
    void    clear();
    T       get(int index);
    void    set(int index, T e);
    int     indexOf(T item);
    bool    contains(T item);
    T       removeAt(int index);
    bool    removeItem(T item);
```

```
    void    ensureCapacity(int index);
```

```
};
```

For example:

Test	Result

ArrayList<int> arr;
 	
for (int i = 0; i < 10; ++i) {
arr.add(i);
}
arr.removeAt(0);
 	
cout << arr.toString() << '\n';
cout << arr.size();
	[1, 2, 3, 4, 5, 6, 7, 8, 9]
	9

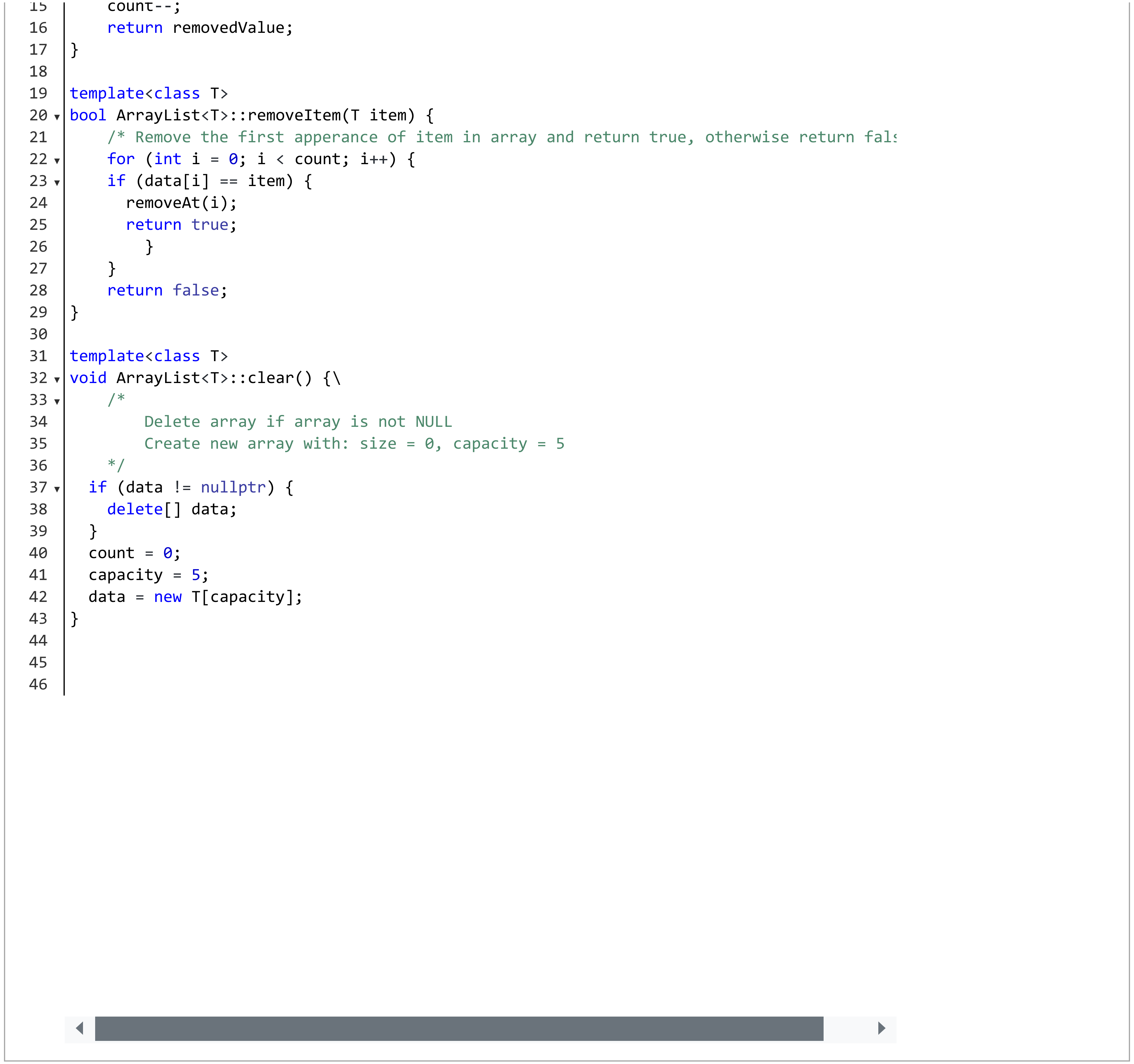
ArrayList<int> arr;
 	
for (int i = 0; i < 10; ++i) {
arr.add(i);
}
arr.removeAt(9);
 	
cout << arr.toString() << '\n';
cout << arr.size();
	[0, 1, 2, 3, 4, 5, 6, 7, 8]
	9

ArrayList<int> arr;
 	
for (int i = 0; i < 10; ++i) {
arr.add(i);
}
arr.removeAt(5);
 	
cout << arr.toString() << '\n';
cout << arr.size();
	[0, 1, 2, 3, 4, 6, 7, 8, 9]
	9

Answer: (penalty regime: 0, 0, 0, 0, 0, 100 %)

Reset answer

```
template<class T>
T ArrayList<T>::removeAt(int index) {
    /*
    Remove element at index and return removed value
    if index is invalid:
    throw std::out_of_range("index is out of range");
    */
    if (index < 0 || index >= count) {
    throw std::out_of_range("Index is out of range");
    }
    T removedValue = std::move(data[index]);
    for (int i = index; i < count - 1; i++) {
    data[i] = std::move(data[i + 1]);
    }
```



```
    count--;
    return removedValue;
}

template<class T>
bool ArrayList<T>::removeItem(T item) {
    /* Remove the first apperance of item in array and return true, otherwise return fals
    for (int i = 0; i < count; i++) {
    if (data[i] == item) {
      removeAt(i);
      return true;
        }
    }
    return false;
}

template<class T>
void ArrayList<T>::clear() {\
    /*
        Delete array if array is not NULL
        Create new array with: size = 0, capacity = 5
    */
  if (data != nullptr) {
    delete[] data;
  }
  count = 0;
  capacity = 5;
  data = new T[capacity];
}



```

	Test	Expected	Got	

✓	`ArrayList<int> arr;`
	
	`    for (int i = 0; i < 10; ++i) {`
	`        arr.add(i);`
	`    }`
	`    arr.removeAt(0);`
	
	`    cout << arr.toString() << '\n';`
	`    cout << arr.size();`
		`[1, 2, 3, 4, 5, 6, 7, 8, 9]`
		`9`
			`[1, 2, 3, 4, 5, 6, 7, 8, 9]`
			`9`
				✓

✓	`    ArrayList<int> arr;`
	
	`    for (int i = 0; i < 10; ++i) {`
	`        arr.add(i);`
	`    }`
	`    arr.removeAt(9);`
	
	`    cout << arr.toString() << '\n';`
	`    cout << arr.size();`
		`[0, 1, 2, 3, 4, 5, 6, 7, 8]`
		`9`
			`[0, 1, 2, 3, 4, 5, 6, 7, 8]`
			`9`
				✓

✓	`    ArrayList<int> arr;`
	
	`    for (int i = 0; i < 10; ++i) {`
	`        arr.add(i);`
	`    }`
	`    arr.removeAt(5);`
	
	`    cout << arr.toString() << '\n';`
	`    cout << arr.size();`
		`[0, 1, 2, 3, 4, 6, 7, 8, 9]`
		`9`
			`[0, 1, 2, 3, 4, 6, 7, 8, 9]`
			`9`
				✓

Passed all tests! ✓

Đúng

Marks for this submission: 1,00/1,00. Accounting for previous tries, this gives **0,00/1,00**.

Câu hỏi **7**

Đúng

Đạt điểm 1,00 trên 1,00

Given an array of integers `nums` and a two-dimension array of integers `operations`.
Each operation in `operations` is represented in the form `{L, R, X}`. When applying an operation, all elements with index in range `[L, R]` (include `L` and `R`) increase by `X`.
Your task is to implement a function with following prototype:

```
vector<int> updateArrayPerRange(vector<int>& nums, vector<vector<int>>& operations);
```

The function returns the array after applying all operation in `operations`.

Note:
- The `iostream`, and `vector` libraries have been included and `namespace std` is being used. No other libraries are allowed.
- You can write helper functions.

For example:

Test	Result

`vector<int> nums {13, 0, 6, 9, 14, 16};`
`vector<vector<int>> operations {{5, 5, 16}, {3, 4, 0}, {0, 2, 8}};`
`printVector(updateArrayPerRange(nums, operations));`
	`[21, 8, 14, 9, 14, 32]`

Answer: (penalty regime: 0 %)

Reset answer

```
vector<int> updateArrayPerRange(vector<int>& nums, vector<vector<int>>& operations) {
  // STUDENT ANSWER
  vector<int> changes(nums.size(), 0);
  for (const auto& op : operations) {
  vector<int>::size_type L = op[0];
  vector<int>::size_type R = op[1];
  int X = op[2];
  changes[L] += X;
  if (R + 1 < nums.size()) {
   changes[R + 1] -= X;
    }
  }
  for (vector<int>::size_type i = 1; i < nums.size(); i++) {
  changes[i] += changes[i - 1];
  }
  for (vector<int>::size_type i = 0; i < nums.size(); i++) {
  nums[i] += changes[i];
  }
  return nums;
}

```

	Test	Expected	Got	

✓	`vector<int> nums {13, 0, 6, 9, 14, 16};`
	`vector<vector<int>> operations {{5, 5, 16}, {3, 4, 0}, {0, 2, 8}};`
	`printVector(updateArrayPerRange(nums, operations));`
		`[21, 8, 14, 9, 14, 32]`	`[21, 8, 14, 9, 14, 32]`	✓

✓	`vector<int> nums {19, 4, 3, 2, 16, 3, 17, 8, 18, 12};`
	`vector<vector<int>> operations {{0, 3, 4}, {2, 5, 12}, {3, 6, 6}, {5, 8, 5}, {8, 9, 8}, {0, 5, 9}, {1, 7, 8}, {1, 1, 3}, {5, 5, 18}};`
	`printVector(updateArrayPerRange(nums, operations));`
		`[32, 28, 36, 41, 51, 61, 36, 21, 31, 20]`	`[32, 28, 36, 41, 51, 61, 36, 21, 31, 20]`	✓

Passed all tests! ✓

Đúng

Marks for this submission: 1,00/1,00.