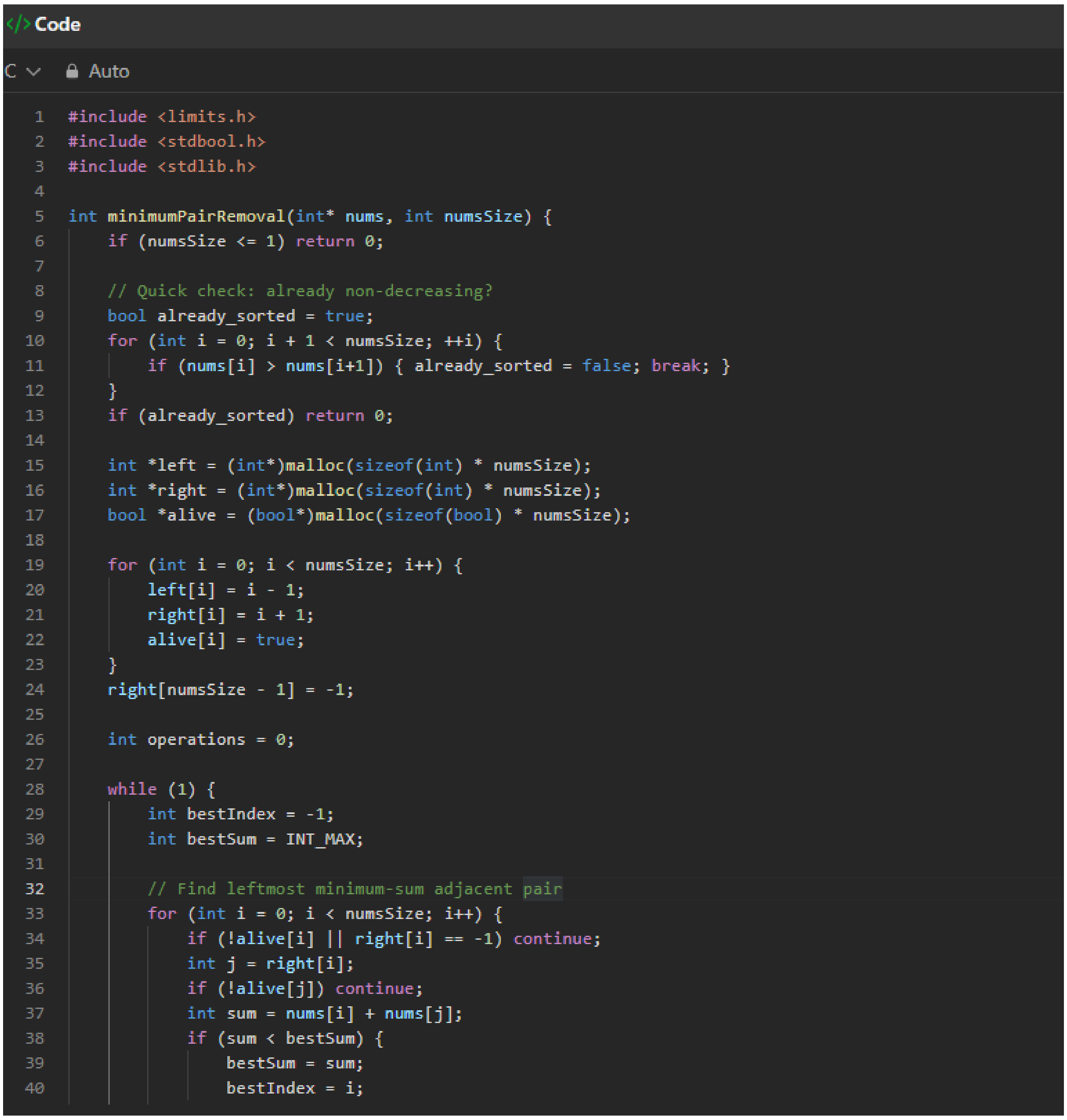
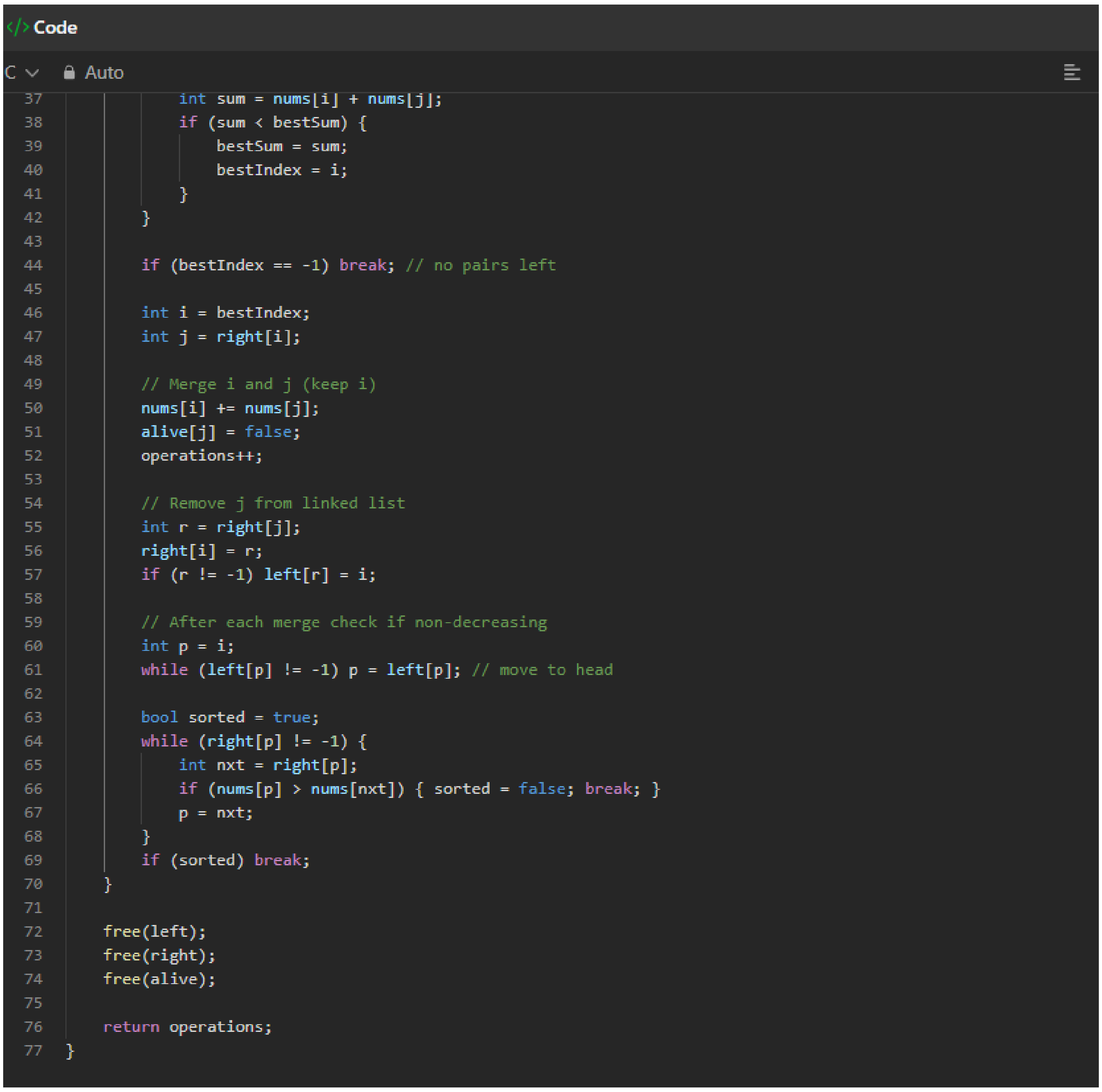
Minimum Pair Removal to Sort Array

Leet Code:



```c
#include <limits.h>
#include <stdbool.h>
#include <stdlib.h>

int minimumPairRemoval(int* nums, int numsSize) {
    if (numsSize <= 1) return 0;

    // Quick check: already non-decreasing?
    bool already_sorted = true;
    for (int i = 0; i + 1 < numsSize; ++i) {
        if (nums[i] > nums[i+1]) { already_sorted = false; break; }
    }
    if (already_sorted) return 0;

    int *left = (int*)malloc(sizeof(int) * numsSize);
    int *right = (int*)malloc(sizeof(int) * numsSize);
    bool *alive = (bool*)malloc(sizeof(bool) * numsSize);

    for (int i = 0; i < numsSize; i++) {
        left[i] = i - 1;
        right[i] = i + 1;
        alive[i] = true;
    }
    right[numsSize - 1] = -1;

    int operations = 0;

    while (1) {
        int bestIndex = -1;
        int bestSum = INT_MAX;

        // Find leftmost minimum-sum adjacent pair
        for (int i = 0; i < numsSize; i++) {
            if (!alive[i] || right[i] == -1) continue;
            int j = right[i];
            if (!alive[j]) continue;
            int sum = nums[i] + nums[j];
            if (sum < bestSum) {
                bestSum = sum;
                bestIndex = i;
```



```c
            int sum = nums[i] + nums[j];
            if (sum < bestSum) {
                bestSum = sum;
                bestIndex = i;
            }
        }

        if (bestIndex == -1) break; // no pairs left

        int i = bestIndex;
        int j = right[i];

        // Merge i and j (keep i)
        nums[i] += nums[j];
        alive[j] = false;
        operations++;

        // Remove j from linked list
        int r = right[j];
        right[i] = r;
        if (r != -1) left[r] = i;

        // After each merge check if non-decreasing
        int p = i;
        while (left[p] != -1) p = left[p]; // move to head

        bool sorted = true;
        while (right[p] != -1) {
            int nxt = right[p];
            if (nums[p] > nums[nxt]) { sorted = false; break; }
            p = nxt;
        }
        if (sorted) break;
    }

    free(left);
    free(right);
    free(alive);

    return operations;
}
```

Input nums = [1,2,2]

Output: 2