#include <iostream>

#include <vector>

#include <unordered\_map>

using namespace std;

vector<int> twoSum(vector<int>& nums, int target) {

unordered\_map<int, int> hash\_map;

vector<int> result;

for (int i = 0; i < nums.size(); ++i) {

int complement = target - nums[i];

// Check if the complement (needed number to achieve target) exists in the hash\_map

if (hash\_map.find(complement) != hash\_map.end()) {

// If found, return the indices

result.push\_back(hash\_map[complement]);

result.push\_back(i);

return result;

}

// Otherwise, store the current number and its index in the hash\_map

hash\_map[nums[i]] = i;

}

// No valid solution found (although problem guarantees exactly one solution)

return result;

}

int main() {

// Example usage:

vector<int> nums1 = {2, 7, 11, 15};

int target1 = 9;

vector<int> result1 = twoSum(nums1, target1);

cout << "Output for Example 1: [" << result1[0] << ", " << result1[1] << "]" << endl;

vector<int> nums2 = {3, 2, 4};

int target2 = 6;

vector<int> result2 = twoSum(nums2, target2);

cout << "Output for Example 2: [" << result2[0] << ", " << result2[1] << "]" << endl;

vector<int> nums3 = {3, 3};

int target3 = 6;

vector<int> result3 = twoSum(nums3, target3);