

Our Solution(s)

Run Code

Your Solutions

Run Code

Solution 1

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 #include <vector>
4 #include <unordered_map>
5 #include <algorithm>
6 using namespace std;
7
8 // Average: O(n^2) time | O(n^2) space
9 // Worst: O(n^3) time | O(n^2) space
10 vector<vector<int>>> fourNumberSum(vector<int> array, int targetSum) {
11     unordered_map<int, vector<vector<int>>>> allPairSums;
12     vector<vector<int>>> quadruplets{};
13     for (int i = 1; i < array.size() - 1; i++) {
14         for (int j = i + 1; j < array.size(); j++) {
15             int currentSum = array[i] + array[j];
16             int difference = targetSum - currentSum;
17             if (allPairSums.find(difference) != allPairSums.end()) {
18                 for (vector<int> pair : allPairSums[difference]) {
19                     pair.push_back(array[i]);
20                     pair.push_back(array[j]);
21                     quadruplets.push_back(pair);
22                 }
23             }
24         }
25         for (int k = 0; k < i; k++) {
26             int currentSum = array[i] + array[k];
27             if (allPairSums.find(currentSum) == allPairSums.end()) {
28                 allPairSums[currentSum] = vector<vector<int>>>{array[k], array
29             } else {
30                 allPairSums[currentSum].push_back(vector<int>{array[k], array[
31             }
32         }
33     }
```

Solution 1

Solution 2

Solution 3

```
1 #include <vector>
2 using namespace std;
3
4 vector<vector<int>>> fourNumberSum(vector<int> array, int targetSum) {
5     // Write your code here.
6     return {};
7 }
8
```

Our Tests

Custom Output

Submit Code

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 #include <vector>
4 #include <unordered_map>
5 #include <algorithm>
6 using namespace std;
7
8 // Average: O(n^2) time | O(n^2) space
9 // Worst: O(n^3) time | O(n^2) space
10 vector<vector<int>>> fourNumberSum(vector<int> array, int targetSum) {
11     unordered_map<int, vector<vector<int>>>> allPairSums;
12     vector<vector<int>>> quadruplets{};
13     for (int i = 1; i < array.size() - 1; i++) {
14         for (int j = i + 1; j < array.size(); j++) {
15             int currentSum = array[i] + array[j];
16             int difference = targetSum - currentSum;
17             if (allPairSums.find(difference) != allPairSums.end()) {
18                 for (vector<int> pair : allPairSums[difference]) {
19                     pair.push_back(array[i]);
20                     pair.push_back(array[j]);
21                     quadruplets.push_back(pair);
22                 }
23             }
24         }
25         for (int k = 0; k < i; k++) {
26             int currentSum = array[i] + array[k];
27             if (allPairSums.find(currentSum) == allPairSums.end()) {
28                 allPairSums[currentSum] = vector<vector<int>>>{array[k], array
29             } else {
30                 allPairSums[currentSum].push_back(vector<int>{array[k], array[
31             }
32         }
33     }
```

```

18     return s
19 }
20
21 bool isPalindrome(string s) {
22     string str = s;
23     reverse(str.begin(), str.end());
24     return str == s;
25 }
26
27 int main() {
28     string s;
29     cin >> s;
30     if (isPalindrome(s))
31         cout << "Yes\n";
32     else
33         cout << "No\n";
34     return 0;
35 }

```

Run or submit code when you're ready.