```java
/*Author: Bochen (mddboc@foxmail.com)
Last Modified: Tue Apr 10 22:28:44 CST 2018*/

/*Given an array S of n integers, are there elements a, b, c, and d in S such that a + b + c + d = target? Find all unique quadruplets in the array which gives the sum of target.

        Note: The solution set must not contain duplicate quadruplets.

        For example, given array S = [1, 0, -1, 0, -2, 2], and target = 0.

        A solution set is:
        [
        [-1,  0, 0, 1],
        [-2, -1, 1, 2],
        [-2,  0, 0, 2]
        ]*/

import java.util.*;
import java.lang.Math;
import java.lang.System;
import java.lang.Integer;


public class Main {

    public static void main(String[] args) {
        int[] nums = {5, 5, 3, 5, 1, -5, 1, -2};

        Solution solution = new Solution();
        List<List<Integer>> receive = solution.fourSum(nums, 4);


        System.out.println("haha");
    }

}


class Solution {
    public List<List<Integer>> fourSum(int[] nums, int target) {

        List<List<Integer>> result = new LinkedList<List<Integer>>();

        if (nums == null || nums.length < 4) {
            return result;
        }

        Arrays.sort(nums);


        int numsLength = nums.length;
        int startPointer, endPointer;
        int sum;
        for (int i = 0; i < numsLength - 3; i++) {

            if (i != 0 && nums[i] == nums[i - 1]) {
                continue;
            }

            for (int j = i + 1; j < numsLength - 2; j++) {

                if (j != i + 1 && nums[j] == nums[j - 1]) {
                    continue;
                }

                if (nums[i] + nums[j] + nums[numsLength - 1] + nums[numsLength - 2] < target) {
                    continue;
                }

                startPointer = j + 1;
                endPointer = numsLength - 1;

                while (startPointer < endPointer) {
                    if (startPointer != j + 1) {
                        while (nums[startPointer] == nums[startPointer - 1]) {
                            startPointer++;
                        }
                    }
                    if (endPointer != numsLength - 1) {
                        while (nums[endPointer] == nums[endPointer + 1]) {
                            endPointer--;
                        }
                    }
                    if (startPointer >= endPointer) {
                        break;
                    }

                    sum = nums[i] + nums[j] + nums[startPointer] + nums[endPointer];
                    if (sum < target) {
                        startPointer++;
                    } else if (sum > target) {
                        endPointer--;
                    } else {
                        result.add(putRightResult(nums[i], nums[j], nums[startPointer], nums[endPointer]));
                        startPointer++;
                        endPointer--;
                    }
                }
            }


        }

        return result;
    }

    private List<Integer> putRightResult(int num1, int num2, int num3, int num4) {
        List<Integer> result = new LinkedList<Integer>();
        result.add(num1);
        result.add(num2);
        result.add(num3);
        result.add(num4);

        return result;
    }
}
```