Target Sum

The given array is not sorted. The given array may or may not contain duplicate elements. Then take the **target** as an integer input. Return Pair of **target sum** in which all pairs are **unique**, for example : `[6, 7], [7, 6]` are considered as the same pair.

Also if the array has repeated elements then return only unique pairs, for eg : if array is `arr = [3, 3, 5, 5]`, and the `target = 8` then result will have only one pair, i.e. `[3, 5]`.

Note : Print the pairs such the smallest integers comes first.

For example `arr = [ 3 , 3, 2, 4]`

output should be:

```
2 4
3 3
```

output should not be:

```
3 3
2 4
```

Sample Input 0

```
4
3 3 5 5
8
```

Sample Output 0

```
3 5
```

1. $\rightarrow$ sort.

tar = 8

3	2	2	5	5	6	7	3	5
0	1	2	3	4	5	6	7	8

2 2 3 3 5 5 5 6 6 7

j i

2 6

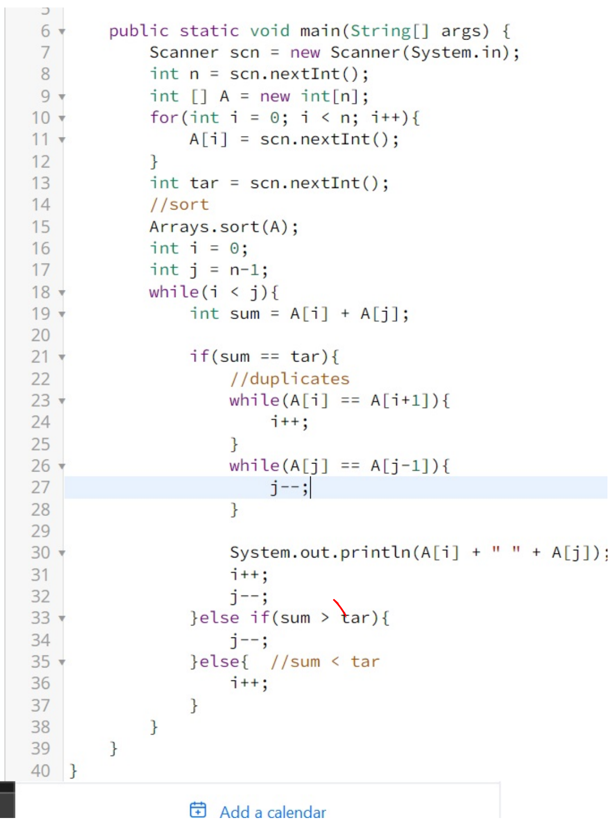
3 5.

sum = 9 ↑ 8 j--

sum = 8 == 8

sum = 8 == 8

tar =8

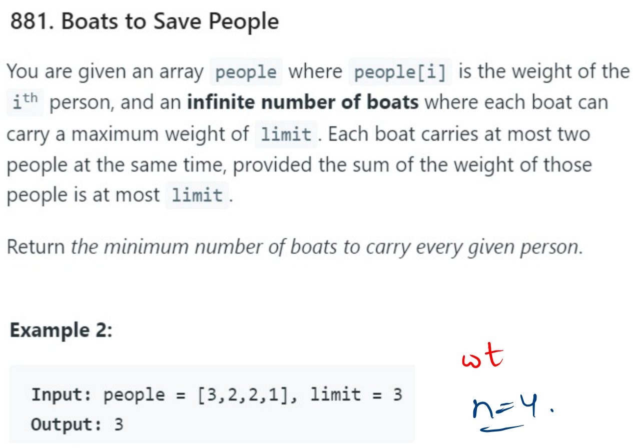


```
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int [] A = new int[n];
        for(int i = 0; i < n; i++){
            A[i] = scn.nextInt();
        }
        int tar = scn.nextInt();
        //sort
        Arrays.sort(A);
        int i = 0;
        int j = n-1;
        while(i < j){
            int sum = A[i] + A[j];

            if(sum == tar){
                //duplicates
                while(A[i] == A[i+1]){
                    i++;
                }
                while(A[j] == A[j-1]){
                    j--;
                }

                System.out.println(A[i] + " " + A[j]);
                i++;
                j--;
            }else if(sum > tar){
                j--;
            }else{  //sum < tar
                i++;
            }
        }
    }
}
```

2 2 3 3 5 5 5 6 6 7

i j

sum = 9



881. Boats to Save People

You are given an array `people` where `people[i]` is the weight of the `i`th person, and an **infinite number of boats** where each boat can carry a maximum weight of `limit`. Each boat carries at most two people at the same time, provided the sum of the weight of those people is at most `limit`.

Return *the minimum number of boats to carry every given person.*

Example 2:

```
Input: people = [3,2,2,1], limit = 3
Output: 3
```

1. ∨ boat at max 2

2. total wt. ≤ limit

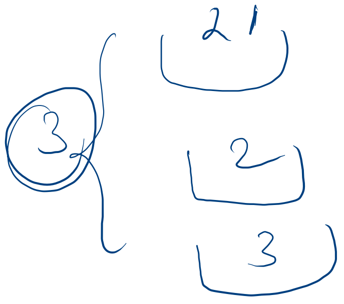
limit=3 ✓

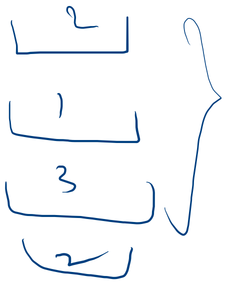
limt ≥ 2 ≥ 1 ≥ 3 ≥ 2 $\boxed{3 \geq 5}$ ✗

wt

n=4.

2 1 3 2 $\boxed{5}$





~~#~~ 4

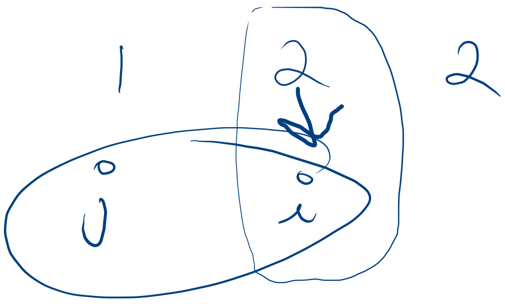
$n = 4$

$\text{limit} = \underline{3}.$

2 3 1 2

$\downarrow$

$\text{ans} = \cancel{0}\ \cancel{1}\ \cancel{2}\ \underline{3}$

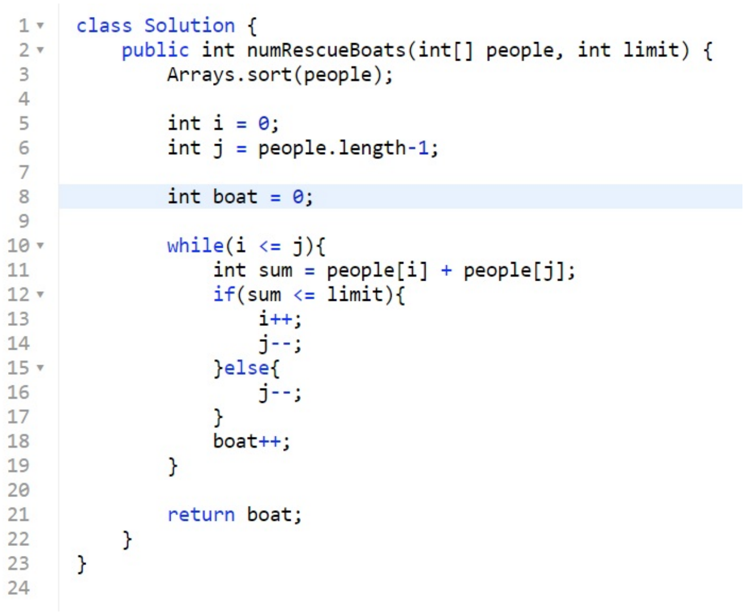


sum = 2 + 1

sum = 4

$\boxed{i > j} \rightarrow \text{stop}$

$\boxed{4 \geq 3}$



```java
class Solution {
    public int numRescueBoats(int[] people, int limit) {
        Arrays.sort(people);

        int i = 0;
        int j = people.length-1;

        int boat = 0;

        while(i <= j){
            int sum = people[i] + people[j];
            if(sum <= limit){
                i++;
                j--;
            }else{
                j--;
            }
            boat++;
        }

        return boat;
    }
}

```

3 Sum

Take an integer array arr as input and print all the triplets `[arr[i], arr[j], arr[k]]` such that `i != j`, `i != k`, and `j != k`, and `arr[i] + arr[j] + arr[k] == 0`.

Notice that the solution set must not contain duplicate triplets.

Sample Input 0

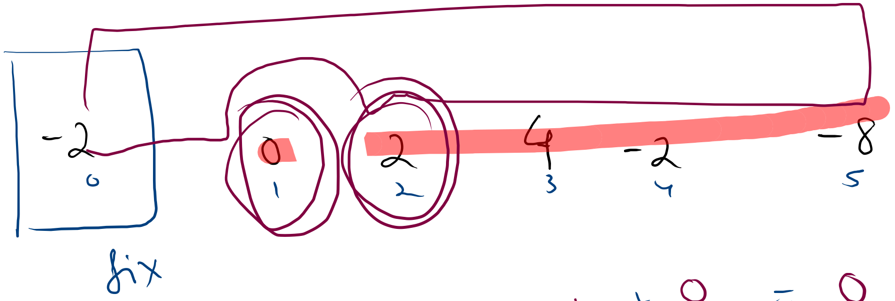
```
6
-2 0 2 4 -2 -8
```

Sample Output 0

```
-2 -2 4
-2 0 2
```

tar = 0

ntar = 2



$x + y + 0 = 0$

$x + y = 0$

$x + y + 2 = 0$ $\quad x + y = -2$

```
-2   -2   4
      0   2
-2
```

$-2 \quad 0 \quad 2 \quad 4 \quad -2 \quad -8$

sorted

$-8 \quad -2 \quad -2 \quad 0 \quad 2 \quad 4$

target -7

$k \quad k+1 \quad n-1$

$-7 < -6 \rightarrow -7$

$-7 \quad -6$

reducing

$-2 \quad -2 \quad 4$

$-2 \quad 0 \quad 2$

$-2 \quad -2 \quad -3 \quad -4 \quad -5$

$i \quad j$

$-2 \quad -5$

$$\begin{array}{ccccc} \underline{-2} & -1 & 0 & \overset{\circ}{2} & 4 \\ & & \overset{\circ}{i} & j & \end{array}$$

$\underline{tar = 2}$

$sum = 1$

$sum > tar$

$j--$

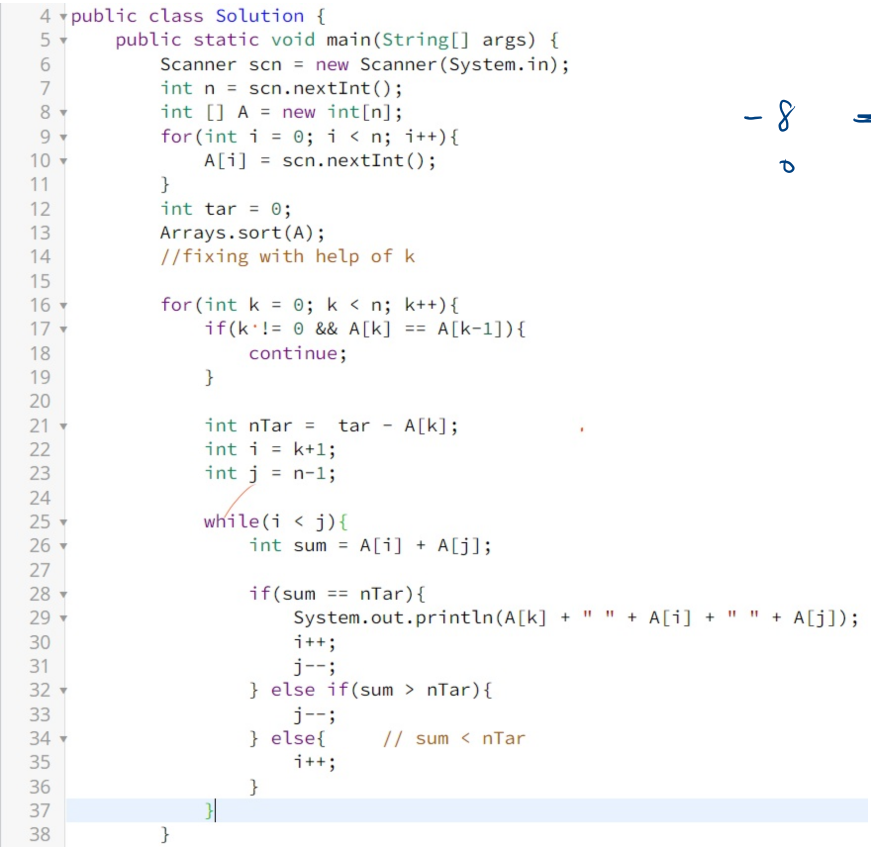
$$\left.\begin{array}{l} \\ \\ \underline{sum < tar} \\ \underline{i++} \end{array}\right\}$$

tar = -1

~~1~~	2	3	4	5	6	7
~~0~~	1	2	3	4	5	6
	i				j	

sum = 9

$$\boxed{\begin{array}{c} \text{sum > tar} \\ \text{i - -} \end{array}}$$



```java
public class Solution {
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int [] A = new int[n];
        for(int i = 0; i < n; i++){
            A[i] = scn.nextInt();
        }
        int tar = 0;
        Arrays.sort(A);
        //fixing with help of k

        for(int k = 0; k < n; k++){
            if(k != 0 && A[k] == A[k-1]){
                continue;
            }

            int nTar =  tar - A[k];
            int i = k+1;
            int j = n-1;

            while(i < j){
                int sum = A[i] + A[j];

                if(sum == nTar){
                    System.out.println(A[k] + " " + A[i] + " " + A[j]);
                    i++;
                    j--;
                } else if(sum > nTar){
                    j--;
                } else{     // sum < nTar
                    i++;
                }
            }
        }
```

ntar = 8

tar = 0

-8	-2	-2	0	2	4
0	1	2	3	4	5

~~k~~

i

k

j

$$\frac{-2 \quad -2 \quad 4}{-2 \quad 0 \quad 2}$$