Date : __ / __ / ____

27 May :

Problem 1 : Operation Sea Evac - The Last Boat from Seagull Island .

→ each boat can carry at most two people .
→ The combined weight must not exceed limit
→ No person can be left behind .

I/P : N = 4
limit = 100
w = [70, 50, 80, 50]
O/P : 3

Approach : Greedy + Two Pointer :
→ Sort the people by weight
→ use two pointer → one from lightest (left),
one from heaviest (right).
→ Pair lightest with heaviest , & put both on boat
if not, heaviest go all alone
→ Move pointer accordingly & increment boat count

Code :
```
public static int minBoatRequired (int N, int limit, int[] w) {
        Arrays.sort(w) ;
        int left =0, right = N-1, boats = 0;
      while ( left <= right) {
         if( w[left] + w[right] <= limit) {
              left ++; }
               right --;
               boat ++;
         }
        return boats ;
        }.
```

Date : __ /

Dry run:

[50, 50, 70, 80]

left = 0 (50)

right = 3 (80)

boats = 0.

i) 50 + 80 = 130 > 100. can't pair

Boat with 80 only.

right -= 1 ⇒ right = 2.

$\boxed{\text{boat} = 1}$

ii) 50 + 70 = 120 > 100 can't pair

go 70 only

right -= 1 ⇒ right = 1. $\boxed{\text{boat} = 2}$

iii) 50+50 = 100 ≤ 100 can pair

Both with go

$\boxed{\text{Boat} = 3}$

Total boats = 3.

2. Festival Fallout → Saving Seats in the Rain.

→ A group must be assigned a full row.
→ Groups cannot be split.
→ You may reorder the groups to seat them efficiently.

I/P: N = 3
C = 5
G = [3, 5, 4]
O/P: 3.

Approach: → sort groups in descending order if combining was allowed.
→ but not allowed.
→ Each group takes one row.

Code:
```
public static int minRowsRequired ( int N, int C, int[] G) {
        return N;
    }
```

Dry Run:
Group 1 → size (3) → 1 row
Group 2 → size (5) → 1 row
Group 3 → size (4) → 1 row