Question: The Minimum Boat Requiered.

n → No. of person (Boat can carry 2 person only)
limit → Max boat can carry.
W[i] → list of weight.

input: N = 4 limit = 100
· W[.] = [70, 50, 80, 50]

Approach:

- Sort weight in ascending Order.
- Use two pointer technique.
- If sum 2 person =< limit pair them in one boat.
- Otherwise 1 boat to havier person
- Count no. of boats.

Pseudocode:

```
→   func. min( weight, limit)
        Sort(weight)
    i=0
    j= len(weight) -1
    Boat = 0

    while ( i <= j )

      if ( wight [i] + weight [j] <= limit)
          i += 1;
          j -= 1;
      boat += 1
      return boat;
```

Dry run:

Input: N=4, limit=100, W=[70,50,80,50]

After sorting: [50, 50, 70, 80]

i=0   j=3

→ Step 1: 50+80 > 100 → j-- → Boat=1

→ Step 2: 50+70 > 100 → j-- → Boat=2

→ step3: 50+50 = 100 → $\boxed{\text{Boat} = 3}$

Question Minimum chair needed or left

Input: N = 3 C = 5 G = [3, 4, 5]

Output: 3.

Approach:

- We will iterati i at every index of G.
- Compare or substract curr no. of chair with maximum i.e C and this wil be our required output.
- We will return left chair.

Pseudocode:

```
func. minchair( groups, c)
    left chairs = 0
for(i -> 0 to n.)

    leftchairs += C - groups[i]

return leftchairs.
```

Dryrun:

input ⇒ N = 3 C = 5 G = [3, 4, 5]

→ Groupp 3 → Row1 (2 empty)
→ Group 4 → Row2 (1 empty)
→ Group 5 → Row 3 (0 empty)

Total liftchair = 3