May 29

Problem 1: Drone Delivery Dilemma - Max packages on Battery.

I/P: N = 4 , B = 10
U = [4, 2, 6, 3]
P = [20, 10, 40, 30]

O/P: 60

⇒ Approach -: • Sort them in descending order
• than track the battery usage with each max profit.
• select the package till the battery usage is less than or equal to 10.

pseudo-code:-

```
Arrays.sort(S)
List<Package> packages = new ArrayList<>();
for(int i=0; i<N; i++)
{   packages.add(new, Package(U[i], P[i]));
}
packag.sort((b,b) -> Double.compare(b.ratio,
a.ratio));
int totalprofit = 0;
for(package pkg : packages)
{   if(B >= pkg.usage){ B -= pkg.usage
    totalprofit += pkg.profit;
}
return total profit;
```

Dry-Run: Sort: P[40, 30, 20, 10] U=[6, 3, 4, 2] B=10

P U

Now, we can take 40 6 ✓ (10 > 6)

30 6+3=9 ✓ (10 > 9)

20 9+4=11 ✗ (10 < 11)

30, 40+30 = 70 profit

Problem 2: Scholarship Distribution - Help the Neediest first.

I/P: N=6, B=50

S = [10, 20, 5, 30, 15, 25]

O/P: 4

⇒ Approach:- Sort requests S[i] in ascending order.

- ~~Feed~~ give the full scholarship from smallest to largest, until total > 50

pseudo code:

```
Arrays.sort(S);
int count = 0;
for (int i = 0; i < N; i++)
{
    if (B >= S[i]){
        B -= S[i];
        count++;
    } else { Break;
    }
return count;
```

Dry-Run:

Sort: S = [5, 10, 15, 20, 25, 30]

Give 5 → Remaining: 45; count = 1

Give 10 → " 35; count = 2

Give 15 → " 20; count = 3

Give 20 → " 0; count = 4

Give 25 → " as Remaining is 0 is can't distributes more

Maximum = 4