DAA

UNIT-2

R Nithin.
192210164.

Assignment-2

① A Conveyor belt has package that must be shipped from one port to another within days days. The ith package on the conveyor belt has a weight of weights. Each day, we load the ship with the packages on the conveyor belt. we May not load more weight than the Maximum weight capacity of the ship. Return the least weight capacity of the belt being shipped within days days.

To find the least weight capacity of the ship that will result in all the packages on the Conveyor belt being shipped.

```
def shipWithinDAYS :
    def is_feasible :
        days_needed = 1
        CURRent_load = 0
    for weight in weights :
        if Current_load + weight > Capacity :
            days_needed += 1
            current_load = 0
        return days_needed <= days
    left, right = max(weight), sum(weights)
    While left < right :
        mid = left + (right - left)//2
        If is_feasible(mid) :
            right = mid
        else :
            left = mid + 1
    return left.

#Example usage :
weights = [1,2,3,4,5,6,7,8,9,10]
days = 5
result = ShipWithinDays(weights, days)
Print (result).
```

This function 'shipwithinDays', uses a binary search algorithm to find the minimum weight capacity of the ship.

② You have n tasks and m workers. Each tasks has strength requirment stored in a 0-indexed integer array tasks, with the task requring tasks strength to complete. the strength of each workers can only be assigned to a single task and must have. Given the 0-indexed imager arrays tasks and workers and the integers pills and strength, return and maximum number of taskes that can be completed.

To Solve this problem, You can use a greedy algorithm.

```
def Maxtasks Completed :
tasks . Sort (reverse = True) .
workers . Sort (reverse = True)
  Completed _ tasks = 0
for task - Strength in tasks :
      assigned = false .
for i, worker_strength in enumerate :
    if worker _ Strength >= task - Strength :
          assigned = True .
          workers . pop (i)
          break
   If not assigned and pills > 0 and workers
and workers [-i] + Strength >= task _ Strength :
          Pills = 1
          workers . pop ()
   If assigned or (not assigned) :
        Completed _ tasks += 1
   return Completed _ tasks :
# Ex usage :-
tasks = [3, 7, 2]
workers = [5, 10, 6]
pills = 2
strength = 2
result = Max Tasks Completed (tasks, workers, pills,
, strength)
print (result)
```

This function, 'max tasks Completed', takes the tasks, workers, pills, and strength as input and returns the Maximum number of tasks that Can be Completed .

③ you have two fruit baskets Containing n fruits each you are given two 0 - Indexed integer arrays basket 2 representing the Cost of fruits in each basket . You want to make both baskets Equal . To do . So You Can use the followings operation as Many times as you want : chose two indices . Return the Minimom Cost to Make both the baskets equal or -1 if impossible .

To Solve this problem, You Can Iterate through all possible swaps and Calculate the Cost of each swaps .

```
def minCost Equial Baskets :
   n = len (basket1)
   Total-Cost = sum(basket1) + sum(basket2)
     if n % 2 == 1 :
        return -1 .
```

```
    half_n = n//2
    basket1. Sort()
    basket 2 .Sort()
min_Cost = float('inf')
  for i in range(half,n):
      Cost = min(basket1[i], basket 2[i])
      min-Cost = min(min_Cost, Cost)
 return total_Cost - 2* main_Cost
```

Ex

```
* basket1 = [1,4,3,5]
  basket2 = [7,9,2,1]
 result = minCostEqualBaskets
 Print(result)
```

The function, 'mincostEqualbaskets', takes two arrays, 'basket1' and 'basket2' as input and returns the minimum Cost to make both baskets Equal.

4 You have n super. washing Machines on a line Initially each washing Machine has some dresses or is empty m(1<=m<=n) machine at the same time. Given an integer array machines representing the number of dresses in each washing Machine from left to right or the line. If it is not possible to do it return-1.

To Solve this problem, You Can Calculate the Cumulative sum of dresses in each washing Machine and determine. the target number of dresses that each machine Should have for the entire line to be balanced.

```
    def findMinMoves(machines):
        total_dresses = sum(machines)
      n = len(machines)
    if total_dresses % n != 0:
          return -1.
  Target_dress = total_dresses//n
    moves, balance = 0, 0
  for dresses in machines:
      imbalance = dresses - target, dresses
      balance += imbalance.
      moves = max(moves, abs(balance),
    return moves.
```

#Example

```
machines = [1,0,5]
result = findwinmoves(machines)
Print(result)
```

it Calculates the target number of dresses, iterates through the machines, and Calculates the based on the balanced at each Points.