A conveyor belt has packages that must be shipped from one port to another within days. The ith package on the conveyor belt (in the order given by weight). we may not load more weight than the maximum weight capacity of the ship. Return the least weight capacity of the ship that will result in all the packages on the conveyor belt being shipped within days.

To find the least weight capacity of the ship needed to ship all packages within days, you can use a binary search Algorithm

```
def ship-within-days(weights,days):
def is-feasible (capacity):
 required-days = 1
 current-weight = 0
for weights in weights:
  if current-weight+weight > capacity:
        required-days+=1
        current-weight=0
   current-weight+= weight
  return required-days <= days

left, right = max(weights), 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                                  #output is 15
print(ship_within_days(weights, days))
```

print("strong password checker")

c) you have n tasks and m workers. each task has a strength requirement stored in a 0-indexed integer array tasks, with the ith task requiring tasks[i] strength to complete. the strength of each worker is stored in a 0-indexed integer array workers, with the jth worker having workers[j] strength. each worker can only be assigned to single task and must have a strength greater than or equal to the task's strength requirement (i.e., workers[j] >= tasks[i]). Additionally, you have magical pills that will increase a worker's strength by strength. you can decide which worker receive the magical pill

To solve this problem, we can sort both the tasks and workers arrays in descending order. Then for each task, we iterate through the workers from the strongest to the weakest.

```
def maxtasks (tasks, workers, pills, strength):
tasks.sort (reverse = true)
workers.sort (reverse = True)
tasks_completed = 0
 used_pills = 0
for task_strength in tasks:
for i, worker_strength in enumerate (workers):
  if worker_strength >= task_strength:
    workers.pop (i).
   tasks_completed += 1
   break
 elif pills > 0 and worker_strength + strength >= task_strength:
   pills -= 1
   used_pills += 1
   tasks_completed += 1
    break
return tasks_completed
tasks = [5, 4, 2, 1]
workers = [7, 3, 2, 1, 5, 6]
pills = 2
strength = 2
print (max tasks (tasks, workers, pills, strength.
```

output should be 4

```
for i in range (1,k+1):
 for j in range (1,n+1):
  for l in range (j,0,-1]):
   dp[i] = max(dp[i], dp[i-1]* max_score[j])
  return dp[k]%MOD
nums = [1,2,3,4]
 k=2
Print(maxprimescore (nums,k)) out put should be 24
```

5) You have two fruit baskets containing n fruits each. you are give two 0-indexed integer arrays basket1 and basket2 representing the cost of fruit in each basket. you want to make both baskets equal. chose two indices i and j, and swap the ith fruit of basket1 with the jth fruit of basket2

Return the minimum cost to make both the baskets equal or -1 if impossible.

```
def min_cost_toequalbaskets (basket1,basket2):
if sorted(basket1) != sorted(basket2):
  return -1
basket1.sort()
basket2.sort()
  n= len(basket1)
  min_cost = 0
for i in range(n):
  min_cost += min(basket1[i], basket2[i])
  return min_cost
Print (mincost_toequalbaskets (basket1,basket2))
```

you have n super washing machines on a line. Initially, each washing machine has some dresses or is empty. For each move, you could choose any m (1<=m<=n) washing machines and pass one dress of each washing machine to one of its adjacent washing machines at the same time.

Given an integer array machines representing the number of dresses in each washing machine from left

```
def minmove to equalDressey (machinw):
   total - dresses = sum (machines)
      n= len( machines
  if total - dresses % n != 0:
      return -1
   Target- dresses = total_dresses//n
     max_moves =0
    dresses_needed = 0
    for dresses in machine:
     dresses - needed += dresses - target - dresses
     max -moves = max (max -moves , abs (dresses -need))
     Print (minmove toequaldresses (machines).
```