5.10 MINIMUM NUMBER OF CONTAINERS USING GREEDY APPROACH

Question:

Given a list of item weights and a maximum capacity for each container, determine the minimum number of containers required to load all items using a greedy approach. The greedy approach should prioritize loading items into the current container until it is full before moving to the next container.

AIM

To implement a greedy algorithm that packs items into containers by filling each container as much as possible before starting a new one.

ALGORITHM

1. Sort the item weights in descending order to pack heavier items first.
2. Initialize a list containers to track remaining capacity in each container.
3. For each item:
    - Try to place it in the first container where it fits.
    - If no container can accommodate it, create a new container.
4. Return the total number of containers used.

PROGRAM

```
def min_containers(weights, capacity):
    weights.sort()
    containers = 0
    i = 0
    while i < len(weights):
        total = 0
        while i < len(weights) and total + weights[i] <= capacity:
            total += weights[i]
            i += 1
        containers += 1
    return containers

n = int(input("Enter number of items: "))
weights = list(map(int, input("Weights: ").split()))
capacity = int(input("Max capacity: "))
print("Minimum containers required:", min_containers(weights, capacity))
```

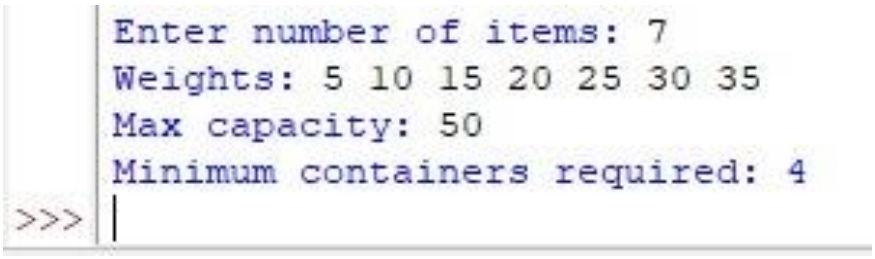
Input:

Enter number of items: 7

Weights: 5 10 15 20 25 30 35

Max capacity: 50

Output:



RESULT:

Thus the program is successfully executed and the output is verified.

PERFORMANCE ANALYSIS:

- Time Complexity: $O(n^2)$ in worst case (due to nested loop), but efficient for small to medium input sizes
- Space Complexity: O(n), for storing container states