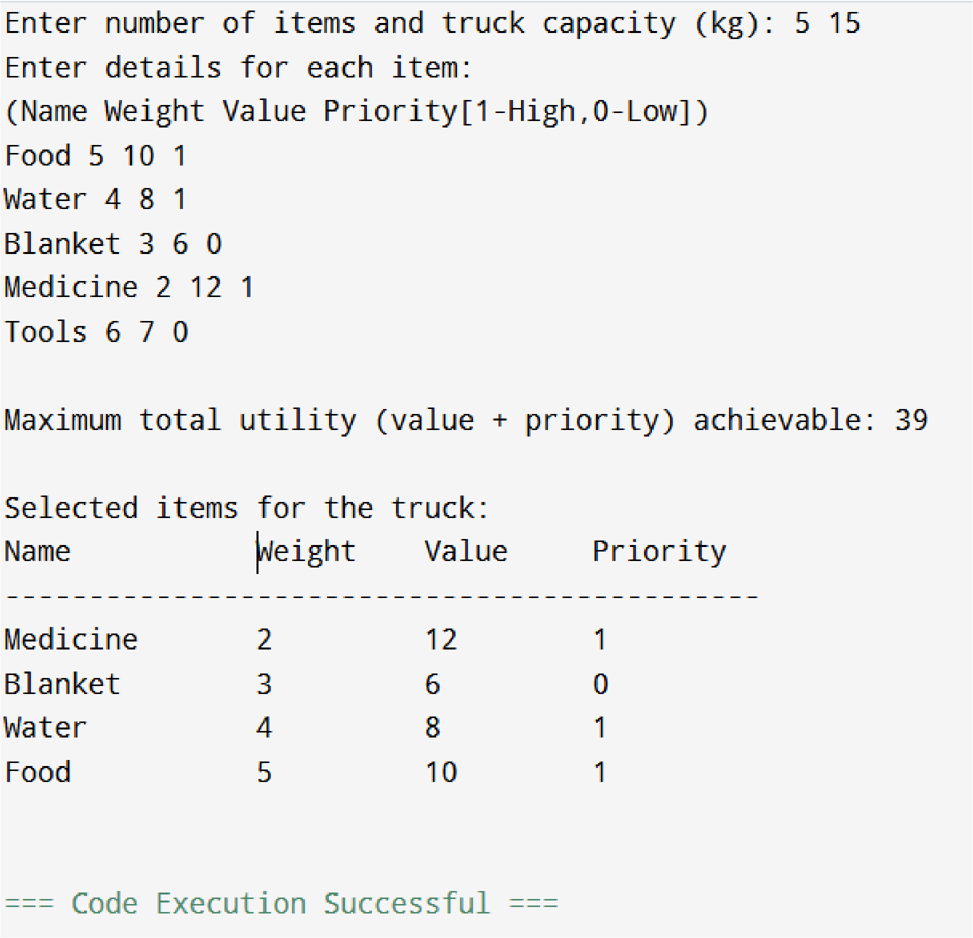
Disaster Relief Resource Allocation Using 0/1 Knapsack

OUTPUT:



```
Enter number of items and truck capacity (kg): 5 15
Enter details for each item:
(Name Weight Value Priority[1-High,0-Low])
Food 5 10 1
Water 4 8 1
Blanket 3 6 0
Medicine 2 12 1
Tools 6 7 0

Maximum total utility (value + priority) achievable: 39

Selected items for the truck:
Name           Weight    Value     Priority
---------------------------------------------
Medicine       2         12        1
Blanket        3         6         0
Water          4         8         1
Food           5         10        1


=== Code Execution Successful ===
```

Input Recap

Item	Weight (kg)	Value	Priority
Food	5	10	1
Water	4	8	1
Blanket	3	6	0
Medicine	2	12	1
Tools	6	7	0

- **Truck capacity**: 15 kg
- **Utility formula**: total_utility = value + priority

Step 1: Calculate Effective Utility

For DP, we use **value + priority** for each item:

Item	Value	Priority	Utility (Value + Priority)
Food	10	1	11
Water	8	1	9
Blanket	6	0	6
Medicine	12	1	13
Tools	7	0	7

Step 2: Optimal Selection (0/1 Knapsack)

- **Truck capacity** = 15 kg.
- DP tries all combinations of items to **maximize total utility** without exceeding 15 kg.

Selected items in output:

Item	Weight	Utility
Medicine	2	13
Blanket	3	6
Water	4	9
Food	5	11

- **Total weight** = 2 + 3 + 4 + 5 = **14 kg** (within 15 kg)
- **Total utility** = 13 + 6 + 9 + 11 = **39**

Why Tools were excluded?

- Tools weight = 6, utility = 7.
- Adding Tools would exceed the optimal combination for maximum utility.
- DP ensures the combination **maximizes utility while staying under weight limit**.

Step 3: Understanding the Order

- The **order of selected items** in output is based on **backtracking through the DP table**, not on weight or priority.
- It lists **all items included** in the optimal solution.

Summary

1. **Maximum total utility** achievable = **39**.
2. **Optimal combination of items** fits in 15 kg:
    - Medicine (2 kg)
    - Blanket (3 kg)
    - Water (4 kg)
    - Food (5 kg)