0/1 Knapsack Problem

The 0/1 Knapsack Problem is a combinatorial optimization problem where you must select items with given values and weights to maximize total value, without exceeding the knapsack's capacity.

Each item can be either taken (1) or not taken (0) — hence the name 0/1 Knapsack (no fractions allowed).

Dynamic Programming Strategy:

1. Create a table dp[i][w] where
   i = number of items considered,
   w = current weight capacity.
2. For each item:
   - If the weight of the item ≤ current capacity,
     take the maximum of:
     - value including the item
     - value excluding the item
3. The final cell dp[n][W] gives the maximum profit.

Complexity:

- Time Complexity: $O(n \times W)$
- Space Complexity: $O(n \times W)$

Code

```
# 0/1 Knapsack Problem using Dynamic Programming

def knapsack(W, wt, val, n):
    dp = [[0 for x in range(W + 1)] for y in range(n + 1)]
    for i in range(n + 1):
        for w in range(W + 1):
            if i == 0 or w == 0:
                dp[i][w] = 0
```

```
            elif wt[i - 1] <= w:
                dp[i][w] = max(val[i - 1] + dp[i - 1][w - wt[i - 1]], dp[i - 1][w])
            else:
                dp[i][w] = dp[i - 1][w]
    return dp[n][W]
# ----- MAIN PROGRAM -----
n = int(input("Enter number of items: "))
val = []
wt = []
for i in range(n):
    v = int(input(f"Enter value of item {i+1}: "))
    w = int(input(f"Enter weight of item {i+1}: "))
    val.append(v)
    wt.append(w)
W = int(input("Enter capacity of knapsack: "))
max_value = knapsack(W, wt, val, n)
print("\nMaximum value that can be obtained =", max_value)
```