Name : Darshan Bele

Div : A , Batch : A1

Practical 4

```
def knapSack(W, wt, val, n):
    # Create DP table K[][]
    K = [[0 for _ in range(W + 1)] for _ in range(n + 1)]

    # Build table K[][] in bottom-up manner
    for i in range(n + 1):
        for w in range(W + 1):
            if i == 0 or w == 0:
                K[i][w] = 0
            elif wt[i - 1] <= w:
                K[i][w] = max(val[i - 1] + K[i - 1][w - wt[i - 1]], K[i - 1][w])
            else:
                K[i][w] = K[i - 1][w]

    # Store result of knapsack
    res = K[n][W]
    w = W
    selected_items = []

    # Trace the selected items
    for i in range(n, 0, -1):
        if res <= 0:
            break
        # If the item was not included
        if res == K[i - 1][w]:
            continue
        else:
            # This item is included.
            selected_items.append(i - 1)
            res -= val[i - 1]
            w -= wt[i - 1]
```

```
    return K[n][W], selected_items


if __name__ == "__main__":
    n = int(input("Enter number of items: "))
    val = list(map(int, input("Enter values of items (space-separated): ").split()))
    wt = list(map(int, input("Enter weights of items (space-separated): ").split()))
    W = int(input("Enter capacity of knapsack: "))

    if len(val) != n or len(wt) != n:
        print("Error: Number of values/weights does not match number of items!")
    else:
        max_value, selected_items = knapSack(W, wt, val, n)

        print("\nMaximum value that can be put in knapsack =", max_value)
        print("Selected items:")

        for i in selected_items[::-1]:  # reverse to maintain original order
            print(f"Item {i+1}: Value = {val[i]}, Weight = {wt[i]}")
```

OUTPUT :

```
PS C:\Users\darsh\Desktop\DAA]>  c:; cd 'c:\Users\darsh\Desktop\DAA]'; & 'c:\Program Files\Python313\python.exe'
'c:\Users\darsh\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '61774' '--'
'c:\Users\darsh\Desktop\DAA]\pract4.py'
Enter number of items: 6
Enter values of items (space-separated): 100 120 80 60 77 86
Enter weights of items (space-separated): 10 20 30 40 50 60
Enter capacity of knapsack: 75

Maximum value that can be put in knapsack = 300
Selected items:
Item 1: Value = 100, Weight = 10
Item 2: Value = 120, Weight = 20
Item 3: Value = 80, Weight = 30
```