```python
def knapsack_tab(val, wt, W):
    n = len(val)

    # Initialize dp array using a nested loop
    dp = []
    for i in range(n + 1):
        row = [0] * (W + 1)  # Create a row of zeros for each item
        dp.append(row)

    # Fill the dp array
    for i in range(1, n + 1):
        for j in range(1, W + 1):
            v = val[i - 1]  # Value of the current item
            w = wt[i - 1]   # Weight of the current item
            if w <= j:
                # Include the current item
                inc_profit = v + dp[i - 1][j - w]
                # Exclude the current item
                exc_profit = dp[i - 1][j]
                # Choose the maximum of including or excluding
                dp[i][j] = max(inc_profit, exc_profit)
            else:
                # Cannot include the current item as its weight
exceeds the capacity
                dp[i][j] = dp[i - 1][j]

    print_dp(dp)  # Print the dp array for visualization
    return dp[n][W]  # Return the maximum value that can be
accommodated in the knapsack

def print_dp(dp):
    # Print the dp array in a readable format
    for row in dp:
        for value in row:
            print(value, end=" ")
        print()  # New line after each row
    print()  # Extra new line after the whole dp table

if __name__ == "__main__":
    val = [15, 14, 10, 45, 30]  # Values of the items
    wt = [2, 5, 1, 3, 4]        # Weights of the items
    W = 7                        # Maximum weight capacity of the
knapsack

    # val = [1, 2, 5, 6]
    # wt = [2, 3, 4, 5]
    # W = 8

    # Print the result of the knapsack_tab function
```

```
    print(f"Maximum Profit by zero-one knapsack is: {knapsack_tab(val,
wt, W)}")

0 0 0 0 0 0 0 0
0 0 15 15 15 15 15 15
0 0 15 15 15 15 15 29
0 10 15 25 25 25 25 29
0 10 15 45 55 60 70 70
0 10 15 45 55 60 70 75

Maximum Profit by zero-one knapsack is: 75
```