108. Knapsack Problem Using Dynamic Programming

PROGRAM:-

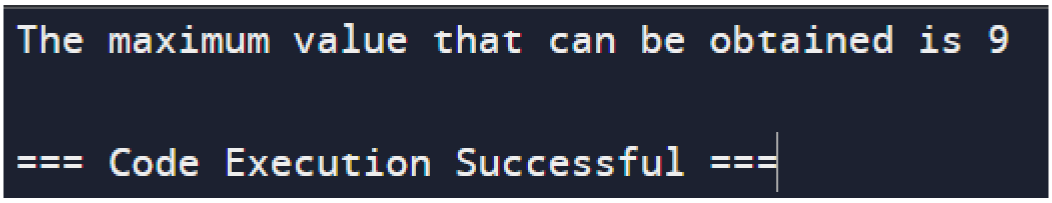
```
def knapsack(weights, values, W):
    n = len(weights)
    # Initialize dp array
    dp = [[0 for _ in range(W + 1)] for _ in range(n + 1)]

    # Build table dp[][] in bottom-up manner
    for i in range(1, n + 1):
        for w in range(1, W + 1):
            if weights[i - 1] <= w:
                dp[i][w] = max(dp[i - 1][w], dp[i - 1][w - weights[i - 1]] + values[i - 1])
            else:
                dp[i][w] = dp[i - 1][w]

    return dp[n][W]

# Example usage:
weights = [1, 3, 4, 5]
values = [1, 4, 5, 7]
W = 7
max_value = knapsack(weights, values, W)
print(f"The maximum value that can be obtained is {max_value}")
```

OUTPUT:-



```
The maximum value that can be obtained is 9


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

TIME COMPLEXITY:-O(n*w)