155. You are given a list of items with their weights and values. Develop a program that utilizes exhaustive search to solve the 0-1 Knapsack Problem. The program should:
Define a function total_value(items, values) that takes a list of selected items (represented by their indices) and the value list as input. It iterates through the selected items and calculates the total value by summing the corresponding values from the value list.

Test Cases:

Simple Case:

Items: 3 (represented by indices 0, 1, 2)
Weights: [2, 3, 1]
Values: [4, 5, 3]
Capacity: 4
Output:

Test Case 1:

Optimal Selection: [0, 2] (Items with indices 0 and 2)
Total Value: 7

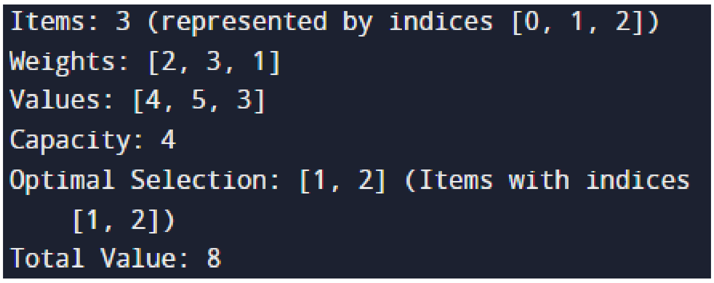
AIM: To solve 0-1 Knapsack problem by utilizing exhaustive search

PROGRAM:

```
import itertools

def total_value(items, values):
    total = 0
    for item in items:
        total += values[item]
    return total

def is_feasible(items, weights, capacity):
    total_weight = sum(weights[item] for item in items)
    return total_weight <= capacity

def knapsack_problem(num_items, weights, values, capacity):
    if num_items == 0:
        return [], 0

    max_value = 0
    optimal_selection = []


    for r in range(num_items + 1):
        for subset in itertools.combinations(range(num_items), r):
            subset_list = list(subset)
            if is_feasible(subset_list, weights, capacity):
                current_value = total_value(subset_list, values)
                if current_value > max_value:
                    max_value = current_value
                    optimal_selection = subset_list

    return optimal_selection, max_value
```

```
def test_knapsack_problem(items, weights, values, capacity, case_name):
    print(f"Test Case {case_name}:")
    print(f"Items: {len(items)} (represented by indices {list(range(len(items)))})")
    print(f"Weights: {weights}")
    print(f"Values: {values}")
    print(f"Capacity: {capacity}")
    optimal_selection, total_val = knapsack_problem(len(items), weights, values, capacity)
    print(f"Optimal Selection: {optimal_selection} (Items with indices {[i for i in optimal_selection]})")
    print(f"Total Value: {total_val}\n")

items1 = [0, 1, 2]
weights1 = [2, 3, 1]
values1 = [4, 5, 3]
capacity1 = 4
test_knapsack_problem(items1, weights1, values1, capacity1, 1)
```

OUTPUT:



```
Items: 3 (represented by indices [0, 1, 2])
Weights: [2, 3, 1]
Values: [4, 5, 3]
Capacity: 4
Optimal Selection: [1, 2] (Items with indices
    [1, 2])
Total Value: 8
```

TIME COMPLEXITY: O(2^n*n)