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:

1. 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.
2. Define a function is_feasible(items, weights, capacity) that takes a list of selected items (represented by their indices), the weight list, and the knapsack capacity as input. It checks if the total weight of the selected items exceeds the capacity.

Program:-

```
import itertools


def total_value(items, values):
    """ Calculate the total value of selected items """
    return sum(values[i] for i in items)


def is_feasible(items, weights, capacity):
    """ Check if selected items are feasible within the capacity """
    total_weight = sum(weights[i] for i in items)
    return total_weight <= capacity


def knapsack_problem(weights, values, capacity):
    """ Solve the 0-1 Knapsack problem using exhaustive search """
    num_items = len(weights)
    best_value = 0
    best_selection = []
    for subset_size in range(1, num_items + 1):
        for subset in itertools.combinations(range(num_items), subset_size):
            if is_feasible(subset, weights, capacity):
                subset_value = total_value(subset, values)
                if subset_value > best_value:
                    best_value = subset_value
                    best_selection = subset
```

return best_selection, best_value

input:-

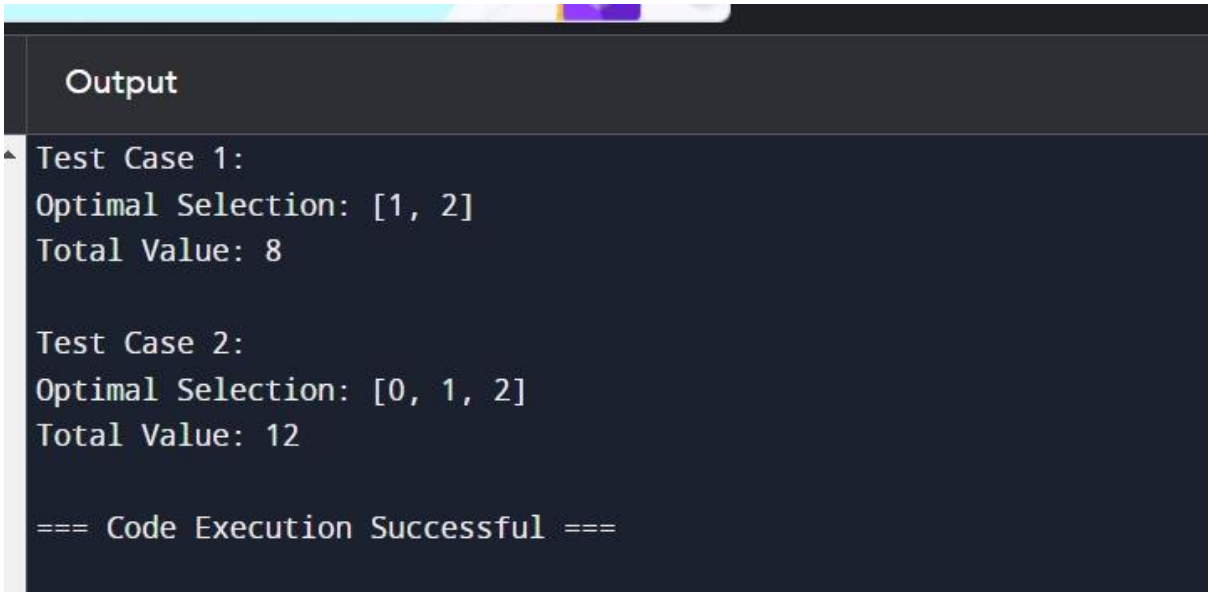
Test cases

weights1 = [2, 3, 1]

values1 = [4, 5, 3]

capacity1 = 4

output:-



TIME COMPLEXITY:-O(2^n)