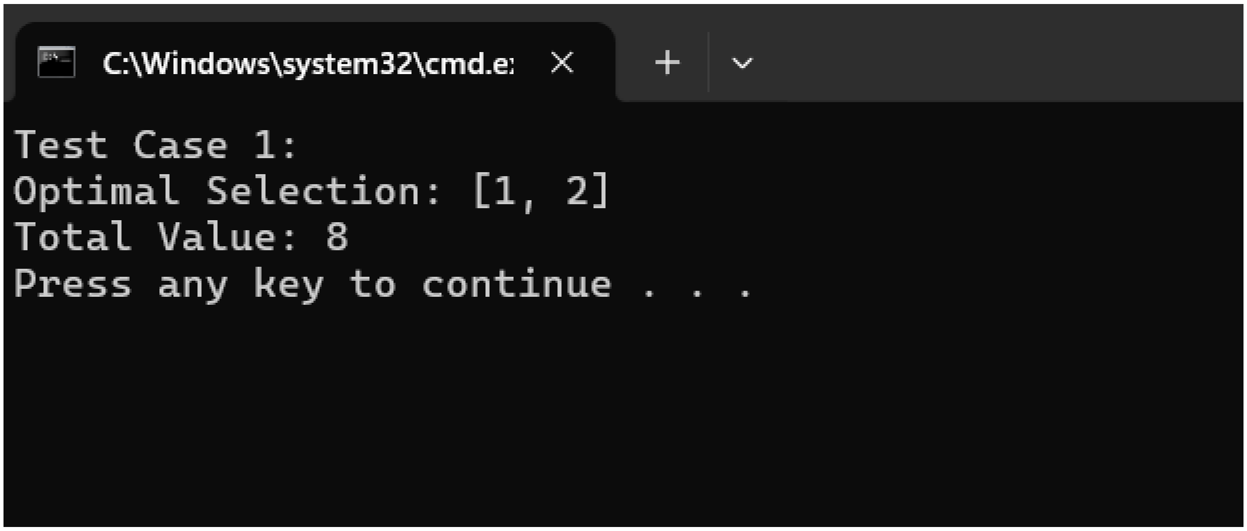
144)Knapsack

CODE:

```
def total_value(items, values):
    return sum(values[i] for i in items)

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

def knapsack_brute_force(values, weights, capacity):
    n = len(values)
    max_value = 0
    optimal_selection = []

    for i in range(2**n):
        selection = [j for j in range(n) if (i & (1 << j))]
        if is_feasible(selection, weights, capacity):
            total = total_value(selection, values)
            if total > max_value:
                max_value = total
                optimal_selection = selection

    return optimal_selection, max_value

# Test Case 1
values1 = [4, 5, 3]
weights1 = [2, 3, 1]
capacity1 = 4
optimal_selection1, total_value1 = knapsack_brute_force(values1, weights1,
capacity1)
print("Test Case 1:")
print("Optimal Selection:", optimal_selection1)
print("Total Value:", total_value1)
```

OUTPUT:



```
Test Case 1:
Optimal Selection: [1, 2]
Total Value: 8
Press any key to continue . . .
```

TIME COMPLEXITY : O(n2*n)