```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

// Structure to represent an item
struct Item {
    int weight;
    int utility;
};

// Function to perform 0/1 Knapsack using Dynamic Programming
int knapsack(const vector<Item>& items, int capacity, vector<int>& selectedItems) {
    int n = items.size();
    vector<vector<int>> dp(n + 1, vector<int>(capacity + 1, 0));

    // Fill DP table
    for (int i = 1; i <= n; ++i) {
        for (int w = 0; w <= capacity; ++w) {
            if (items[i - 1].weight <= w) {
                // Option 1: Include item
                int include = dp[i - 1][w - items[i - 1].weight] + items[i - 1].utility;
                // Option 2: Exclude item
                int exclude = dp[i - 1][w];
                dp[i][w] = max(include, exclude);
            } else {
                dp[i][w] = dp[i - 1][w]; // Item can't be included
            }
        }
    }

    // Trace back to find which items were included
    int remainingWeight = capacity;
    for (int i = n; i > 0; --i) {
        if (dp[i][remainingWeight] != dp[i - 1][remainingWeight]) {
            selectedItems.push_back(i - 1);
            remainingWeight -= items[i - 1].weight;
        }
    }

    return dp[n][capacity];
}

int main() {
```

```
    int n, capacity;

    cout << "Enter the number of items: ";
    cin >> n;
    cout << "Enter the truck capacity: ";
    cin >> capacity;

    vector<Item> items(n);

    cout << "Enter the weight and utility of each item:\n";
    for (int i = 0; i < n; ++i) {
        cout << "Item " << i + 1 << " - Weight: ";
        cin >> items[i].weight;
        cout << "Item " << i + 1 << " - Utility: ";
        cin >> items[i].utility;
    }

    vector<int> selectedItems;
    int maxUtility = knapsack(items, capacity, selectedItems);

    cout << "\nMaximum Utility that can be carried: " << maxUtility << endl;
    cout << "\nItems chosen:\n";

    // Display chosen items
    for (int i : selectedItems) {
        cout << "Item " << i + 1
             << " - Weight: " << items[i].weight
             << ", Utility: " << items[i].utility << endl;
    }

    return 0;
}
```

OUTPUT:

```
Enter the number of items: 4
Enter the truck capacity: 7
Enter the weight and utility of each item:
Item 1 - Weight: 1
Item 1 - Utility: 1
Item 2 - Weight: 3
Item 2 - Utility: 4
Item 3 - Weight: 4
```

Item 3 - Utility: 5
Item 4 - Weight: 5
Item 4 - Utility: 7

Maximum Utility that can be carried: 9

Items chosen:
Item 3 - Weight: 4, Utility: 5
Item 2 - Weight: 3, Utility: 4