```
#include <bits/stdc++.h>
#include <thread>
#include <chrono>

using namespace std;


#define RESET   "\033[0m"
#define RED     "\033[1;31m"
#define GREEN   "\033[1;32m"
#define BLUE    "\033[1;34m"


void clearScreen() {
    cout << "\033[2J\033[H";
}

//  The DP table
void showTable(const vector<vector<int>>& dp, const vector<int>& weights, const vector<int>&
values, int currentItem = -1, int currentCapacity = -1) {
    int itemCount = weights.size();
    int maxCapacity = dp[0].size() - 1;

    clearScreen();
    cout << RED << "Knapsack Table (Rows = Items, Columns = Capacity):\n" << RESET <<
endl;


    cout << setw(13) << "Item/Cap";
    for (int c = 0; c <= maxCapacity; ++c) {
        cout << BLUE << setw(6) << c << RESET;
    }
    cout << "\n" << setw(13) << " ";
    for (int c = 0; c <= maxCapacity; ++c) cout << "------";
    cout << "\n";


    for (int i = 0; i <= itemCount; ++i) {
        if (i == 0) cout << setw(13) << "None";
        else cout << setw(13) << "Item " + to_string(i) + " (w=" + to_string(weights[i - 1]) + ")";

        for (int c = 0; c <= maxCapacity; ++c) {
            if (i == currentItem && c == currentCapacity && i != 0 && c != 0)
                cout << RED << setw(6) << dp[i][c] << RESET;
```

```
            else
                cout << setw(6) << dp[i][c];
        }
        cout << "\n";
    }

    this_thread::sleep_for(chrono::milliseconds(500));
}

// Main knapsack function
void knapsackSolver(int itemCount, int maxCapacity, vector<int> weights, vector<int> values) {
    vector<vector<int>> dp(itemCount + 1, vector<int>(maxCapacity + 1, 0));

    for (int i = 1; i <= itemCount; ++i) {
        for (int c = 0; c <= maxCapacity; ++c) {
            if (weights[i - 1] <= c) {
                int include = values[i - 1] + dp[i - 1][c - weights[i - 1]];
                int exclude = dp[i - 1][c];
                dp[i][c] = max(include, exclude);
            } else {
                dp[i][c] = dp[i - 1][c];
            }
            showTable(dp, weights, values, i, c);
        }
    }

    clearScreen();
    cout << BLUE << "Final Knapsack Table:" << RESET << endl;
    showTable(dp, weights, values);

    cout << GREEN << "\nMaximum Total Value: " << RESET << dp[itemCount][maxCapacity] <<
"\n";

    // Show which items were selected
    int i = itemCount, c = maxCapacity;
    vector<int> selected;
    while (i > 0 && c >= 0) {
        if (dp[i][c] != dp[i - 1][c]) {
            selected.push_back(i - 1);
            c -= weights[i - 1];
        }
        i--;
    }
```

```
    cout << "\nItems Chosen in the Knapsack:\n";
    cout << left << setw(10) << "Item" << setw(10) << "Weight" << setw(10) << "Value" << "\n";
    cout << "-------------------------------------\n";
    for (int idx : selected) {
        cout << left << setw(10) << ("Item " + to_string(idx + 1))
             << setw(10) << weights[idx]
             << setw(10) << values[idx] << "\n";
    }
}

int main() {
    // Example: 4 items, max capacity 8
    int itemCount = 4;
    int maxCapacity = 8;
    vector<int> weights = {2, 3, 4, 5};
    vector<int> values = {1, 2, 5, 6};

    knapsackSolver(itemCount, maxCapacity, weights, values);
    getchar();
    return 0;
}
```