using namespace std;

#include <iostream>

#include <vector>

class Knapsack {

public:

virtual int solveKnapsack(const vector<int> &profits, vector<int> &weights, int capacity) {

vector<vector<int>> dp(profits.size(), vector<int>(capacity + 1, -1));

return this->knapsackRecursive(dp, profits, weights, capacity, 0);

}

private:

int knapsackRecursive(vector<vector<int>> &dp, const vector<int> &profits, vector<int> &weights,

int capacity, int currentIndex) {

// base checks

if (capacity <= 0 || currentIndex >= profits.size()) {

return 0;

}

// if we have already solved a similar problem, return the result from memory

if (dp[currentIndex][capacity] != -1) {

return dp[currentIndex][capacity];

}

// recursive call after choosing the element at the currentIndex

// if the weight of the element at currentIndex exceeds the capacity, we shouldn't process this

int profit1 = 0;

if (weights[currentIndex] <= capacity) {

profit1 = profits[currentIndex] + knapsackRecursive(dp, profits, weights,

capacity - weights[currentIndex],

currentIndex + 1);

}

// recursive call after excluding the element at the currentIndex

int profit2 = knapsackRecursive(dp, profits, weights, capacity, currentIndex + 1);

dp[currentIndex][capacity] = max(profit1, profit2);

return dp[currentIndex][capacity];

}

};

int main(int argc, char \*argv[]) {

Knapsack ks;

vector<int> profits = {1, 6, 10, 16};

vector<int> weights = {1, 2, 3, 5};

int maxProfit = ks.solveKnapsack(profits, weights, 7);

cout << "Total knapsack profit ---> " << maxProfit << endl;

maxProfit = ks.solveKnapsack(profits, weights, 6);

cout << "Total knapsack profit ---> " << maxProfit << endl;

}

