**// Define maximum number of items**

**#define MAX\_ITEMS 100**

**// Structure for an item which has weight and value**

**struct Item {**

**int weight;**

**int value;**

**};**

**// Function to find the maximum of two integers**

**int max(int a, int b) {**

**return (a > b) ? a : b;**

**}**

**// Function to solve the 0/1 Knapsack problem**

**void knapsack(int W, struct Item items[], int n) {**

**// Create a 2D array to store the values of subproblems**

**int dp[n+1][W+1];**

**int i, w;**

**// Build DP table in bottom-up manner**

**for (i = 0; i <= n; i++) {**

**for (w = 0; w <= W; w++) {**

**if (i == 0 || w == 0)**

**dp[i][w] = 0;**

**else if (items[i-1].weight <= w)**

**dp[i][w] = max(items[i-1].value + dp[i-1][w - items[i-1].weight], dp[i-1][w]);**

**else**

**dp[i][w] = dp[i-1][w];**

**}**

**}**

**printf("\nw v n ");**

**for(i=0;i<W+1;i++){**

**printf("%d ",i);**

**}**

**for(i=0;i<n+1;i++){**

**printf("\n");**

**if(i==0){**

**printf(" ");**

**}**

**else{**

**printf("%d %d %d ",items[i-1].weight, items[i-1].value, i);**

**}**

**for(w=0;w<W+1;w++){**

**printf("%d ",dp[i][w]);**

**}**

**}**

**printf("\n");**

**// Maximum value that can be obtained**

**int max\_value = dp[n][W];**

**printf("Maximum value: %d\n", max\_value);**

**// Determine the items included in the knapsack**

**int total\_weight = 0;**

**printf("Selected items:\n");**

**for (i = n; i > 0 && max\_value > 0; i--) {**

**if (max\_value != dp[i-1][W]) {**

**printf("Item %d (value = %d, weight = %d)\n", i, items[i-1].value, items[i-1].weight);**

**max\_value -= items[i-1].value;**

**W -= items[i-1].weight;**

**total\_weight += items[i-1].weight;**

**}**

**}**

**printf("Total weight of selected items: %d\n", total\_weight);**

**}**

**int main() {**

**// Example usage**

**int W = 5; // Knapsack capacity**

**struct Item items[] = {{2,12},{1,10},{3,20},{2,15}}; // Array of items (weight, value);;;;**

**int n = sizeof(items) / sizeof(items[0]); // Number of items**

**// Solve the knapsack problem**

**knapsack(W, items, n);**

**return 0;**

**}**

