1. **Optimal cost**

**Code:**

#include <stdio.h>

#define MAX\_ITEMS 100

// Function to find the maximum of two integers

int max(int a, int b) {

return (a > b) ? a : b;

}

// Function to find the maximum profit using dynamic programming

int knapsack(int weights[], int profits[], int n, int capacity) {

int dp[MAX\_ITEMS + 1][capacity + 1];

// Initialize dp table

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

for (int w = 0; w <= capacity; w++) {

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

dp[i][w] = 0;

else if (weights[i - 1] <= w)

dp[i][w] = max(profits[i - 1] + dp[i - 1][w - weights[i - 1]], dp[i - 1][w]);

else

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

}

}

return dp[n][capacity];

}

int main() {

int weights[MAX\_ITEMS], profits[MAX\_ITEMS];

int n, capacity;

// Input the number of items

printf("Enter the number of items: ");

scanf("%d", &n);

// Input weights and profits of each item

printf("Enter weights and profits of each item:\n");

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

printf("Item %d: ", i + 1);

scanf("%d %d", &weights[i], &profits[i]);

}

// Input the capacity of the knapsack

printf("Enter the capacity of the knapsack: ");

scanf("%d", &capacity);

// Find the maximum profit

int maxProfit = knapsack(weights, profits, n, capacity);

printf("Maximum profit: %d\n", maxProfit);

return 0;

}

**Output:**

Enter the number of items: 5

Enter weights and profits of each item:

Item 1: 3 10

Item 2: 3 15

Item 3: 2 10

Item 4: 5 12

Item 5: 1 8

Enter the capacity of the knapsack: 10

Maximum profit: 43

--------------------------------

Process exited after 30.28 seconds with return value 0

Press any key to continue . . .

