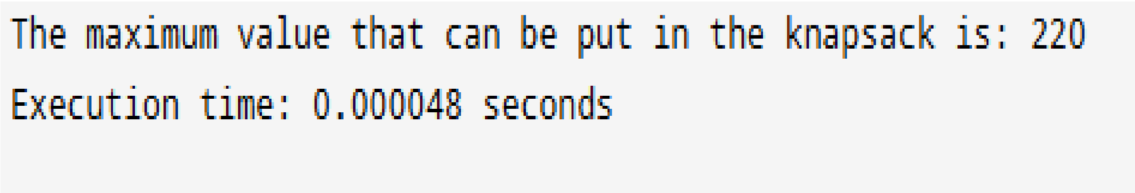
Description:

C program that solves the 0/1 Knapsack problem, and reports execution time using clock().

Source Code:

```
#include <stdio.h>
#include <time.h>

// 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
int knapsack(int W, int wt[], int val[], int n) {
    int i, w;
    // Create a 2D array to store results of subproblems
    // knap[i][w] will store the maximum value for 'i' items and capacity 'w'
    int knap[n + 1][W + 1];

    // Build the knap table in a bottom-up manner
    for (i = 0; i <= n; i++) {
        for (w = 0; w <= W; w++) {
            // Base case: If no items or no capacity, value is 0
            if (i == 0 || w == 0) {
                knap[i][w] = 0;
            } else if (wt[i - 1] <= w) {
                // Take the maximum of two cases:
                // 1. Include the current item: val[i-1] + knap[i-1][w - wt[i-1]]
                // 2. Exclude the current item: knap[i-1][w]
                knap[i][w] = max(val[i - 1] + knap[i - 1][w - wt[i - 1]], knap[i - 1][w]);
            } else {
                // If current item's weight is more than current capacity, exclude it
                knap[i][w] = knap[i - 1][w];
            }
        }
    }
    // The maximum value will be in knap[n][W]
    return knap[n][W];
}

// Driver code to test the knapsack function
int main() {
    clock_t start, end;
    double cpu_time_used;
    int val[] = {60, 100, 120}; // Values of items
    int wt[] = {10, 20, 30};    // Weights of items
    int W = 50;                 // Maximum capacity of the knapsack
    int n = sizeof(val) / sizeof(val[0]); // Number of items

    start = clock();
    printf("The maximum value that can be put in the knapsack is: %d\n", knapsack(W, wt, val, n));
    end = clock();


    cpu_time_used = ((double)(end - start) / CLOCKS_PER_SEC);
    printf("Execution time: %f seconds\n", cpu_time_used);


    return 0;
}
```

Output:



```
The maximum value that can be put in the knapsack is: 220
Execution time: 0.000048 seconds
```