```cpp
//
//   There are N items, numbered 1,2, … N. For each i (1 ⩽ i ⩽ N),
//   Item i has a weight of w[i] and a value of v[i]
//   Taro has decided to choose some of the N items and carry them home in a knapsack.
//   The capacity of the knapsack is W, which means that the sum of the weights
//   of items taken must be at most W.
//
//   Find the maximum possible sum of the values of items that Taro takes home.
//
//   Constraints:
//     1 ⩽ N ⩽ 100
//     1 ⩽ W ⩽ 10^9
//     1 ⩽ w[i] ⩽ W
//     1 ⩽ v[i] ⩽ 10^3
//
//   Time Complexity: O(N * max(v[i]))
//

#include <bits/stdc++.h>
#define ll long long

using namespace std;

int N, W;
vector<int> weights, values;
vector<vector<ll>> dp(101, vector<ll>(100001, -1));

// returns the weight
ll capacity(int i, int value) {
    if (value ⩽ 0) return 0;
    if (i == N) return 1e12;
    if (dp[i][value] ≠ -1) return dp[i][value];

    dp[i][value] = min(
        capacity(i+1, value),
        capacity(i+1, value - values[i]) + weights[i]
    );

    return dp[i][value];
}

int main() {
    cin >> N >> W;

    weights.resize(N);
    values.resize(N);

    int sum_values = 0;
    for (int i = 0; i < N; i++) {
        cin >> weights[i] >> values[i];
        sum_values += values[i];
    }

    for (int value = sum_values; value ⩾ 1; value--) {
        if (capacity(0, value) ⩽ W) {
            cout << value << endl;
            return 0;
        }
    }

    cout << 0 << endl;
    return 0;
}
```