```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 ≤ 10^5
//     1 ≤ w[i] ≤ W
//     1 ≤ v[i] ≤ 10^9
//
//   Time Complexity: O(NW)
//


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

using namespace std;

int main() {
    int n, W;
    cin >> n >> W;

    vector<vector<ll>> dp(n+1, vector<ll>(W+1, 0));
    vector<int> weights(n+1), values(n+1);

    for (int i = 1; i ≤ n; i++) {
        cin >> weights[i] >> values[i];
    }

    for (int i = 0; i ≤ n; i++) {
        for (int w = 0; w ≤ W; w++) {
            if (i == 0 || w == 0) dp[i][w] = 0;
            else if (w - weights[i] ≥ 0) {
                // you can take that item
                dp[i][w] = max(dp[i-1][w], dp[i-1][w-weights[i]] + values[i]);
            } else {
                dp[i][w] = dp[i-1][w];
            }
        }
    }

    cout << dp[n][W] << endl;
    return 0;
}
```