```cpp
//
//   There are N blocks, numbered 1,2, ... N. For each i (1 ⩽ i ⩽ N),
//   Block i has a weight of w[i], a solidness of s[i] and a value of v[i].
//   Taro has decided to build a tower by choosing some of the N blocks
//   and stacking them vertically in some order.
//   Here, the tower must satisfy the following condition:
//      - For each Block i contained in the tower,
//        the sum of the weights of the blocks stacked above it is not greater than s[i].
//   Find the maximum possible sum of the values of the blocks contained in the tower.
//
//     Time Complexity: O(N * max(W))
//
//     ! Note: MXW HAS TO BE AT LEAST 2 * max(w[i]) !
//     ! Note: Memory usage can get too big quickly !
//

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

using namespace std;

struct block {
    ll weight, solidness, value;

    bool operator<(const block& other) {
        return (weight + solidness) > (other.weight + other.solidness);
        // we want to exchange the blocks if
        //     w[top] > w[bottom]
        //  &  s[top] > s[bottom] ⇒
        //     w[top] + s[top] > w[bottom] + s[bottom]
    }
};

inline void umax(ll& a, ll b) {
    if (a < b) a = b;
}

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

    vector<block> a(n);
    for (int i = 0; i < n; i++) {
        cin >> a[i].weight >> a[i].solidness >> a[i].value;
    }

    sort(a.begin(), a.end());

    const int mxw = 3e4 + 10;
    vector<vector<ll>> dp(n, vector<ll>(mxw, 0));
    // dp[i][j] - to the i-th box, if j weight remains

    ll ans = 0;

    // Base case
    for (int w = 0; w ⩽ a[0].solidness; w++) {
        umax(dp[0][w], a[0].value);
        umax(ans, dp[0][w]);
    }

    for (int i = 1; i < n; i++) {
        for (int j = 0; j < mxw; j++) {
            umax(dp[i][j], dp[i-1][j]);
            int weight_remains = min(j - a[i].weight, a[i].solidness);
            if (weight_remains ⩾ 0) {
                umax(dp[i][weight_remains], dp[i-1][j] + a[i].value);
                umax(ans, dp[i][weight_remains]);
            }
        }
    }

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