Assignment 4 Problem 1: Garage Sale

Construct a 3D-array named Max_Price[ ][ ][ ], where Max_Price[i][j][k] is the highest price that can be carry with first i items using total weight j and total volume k.
The pseudo code is as follows

$n \coloneqq number\ of\ items$
$p \coloneqq array\ of\ prices$
$w \coloneqq array\ of\ corresponding\ weights$
$v \coloneqq array\ of\ corresponding\ volumes$
$Max\_Price\ initialized\ to\ be\ 0$
$for\ i\ from\ 1\ to\ n\ do$
    $for\ j\ from\ 0\ to\ W\ do$
        $for\ k\ from\ 0\ to\ V\ do$
            $Max\_Price[i][j][k] = Max(Max\_Price[i-1][j][k],$
                $Max\_Price[i-1][j-w[i]][k-v[i]] + p[i])$

Note that for Max_Price[i][j][k], if any of i,j,k are negative integer, then the value is 0. (when writing code, we can add some if statements to achieve this since the index cannot be negative)
By running the nested loop, for each i,j,k, we can find the highest price that can be carry with first i items using total weight j and total volume k.
This is because for the ith item, we have two options: carry or do not carry.
If we choose to carry, then we need the free weight w[i] and the free volume v[i], this time, the maximum price we can carry is the price of the ith item plus the maximum price with first i-1 items using total wight j-w[i] and total volume k-v[i] (i.e. Max_Price[i-1][j-w[i]][k-v[i]])
If we choose not to carry, then the maximum price we can carry is the maximum price with first i-1 items using total wight j and total volume k (i.e. Max_Price[i-1][j][k])
So we will choose the value which is the maximum.
For base cases, since the array is initialized to be 0, no matter how many items we can choose, if we can only use negative volume or weight, then we can bring nothing. (since all values, weights, and volumes are non-negative.)
We can inductively get all values of the array since each index are equal or less than the one's we want.
To output the value that we desire, we only need to output the Max_Price[n][W][V] which indicates the maximum value of first n items (i.e. all items) with total weight W and total volume V.
The running time of the algorithm is obviously O(nVW), since in the most inner loop, the time it take is a constant time.