```cpp
#include<iostream>
#include<string>

using namespace std;

#ifndef _DYNAMIC_H_
#define _DYNAMIC_H_


/**
* Item Declaration
*/
struct item
{
    string name;
    int weight;
    int value;
    double ratio;
};

/**
 * Dynamic Approach to solving the problem
 */
void Dynamic(int numItems, int capacity, int ** matrix, item * items)
{
    for (int i = 0; i <= numItems; i++)
        for (int j = 0; j <= capacity; j++)
            if (i == 0 || j == 0)
                continue;
            else if (items[i - 1].weight <= j)
                matrix[i][j] = max(matrix[i - 1][j], items[i - 1].value + matrix[i -
                1][j - items[i - 1].weight]);
            else
                matrix[i][j] = matrix[i - 1][j];

    return;
}


/**
 * Refined Dynamic Approach to solving the problem
 */
int Refined_Dynamic(int numItems, int capacity, int ** matrix, item * items)
{
        if (numItems == 0 || capacity == 0)
            return 0;
        if(capacity < items[numItems-1].weight)
            matrix[numItems][capacity] = Refined_Dynamic(numItems - 1, capacity,
            matrix, items);
        else
            matrix[numItems][capacity] = max(Refined_Dynamic(numItems - 1, capacity,
            matrix, items), (Refined_Dynamic(numItems - 1, capacity -
            items[numItems-1].weight, matrix, items) + items[numItems-1].value));
}

int max(int a, int b)
{
    cout << "a " << a << "b: " << b << endl;
     if (a >= b)
        return a;
    return b;
}

#endif
```