```cpp
#include <iostream>
#include <fstream>
#include "dynamic.h"

using namespace std;

int main ()
{
    //Open text file for reading
    ifstream file;
    file.open("jouleFile.txt");

    //Read in the number of items and capacity of the knapsack
    int numItems = 0, capacity = 0;
     file >> numItems >> capacity;

    //Read in the items
    item * items = new item [numItems];
    for (int i = 0; i < numItems; i++)
    {
        item knapItem;
        file >> knapItem.name >> knapItem.value >> knapItem.weight;
        knapItem.ratio = (double) knapItem.value/knapItem.ratio;
        items[i] = knapItem;
    }

    //Make a matrix with dimension numItems x capacity
    //and add 1 to each because stupid walking off the edge of array
    int ** matrix = new int * [numItems + 1];
    for (int i = 0; i < numItems + 1; i++)
        matrix[i] = new int [capacity + 1];

    //Set the whole matrix to 0s
    for (int i = 0; i < numItems + 1; i++)
        for (int j = 0; j < capacity + 1; j++)
            matrix[i][j] = 0;

    //Dynamic Programming
    Dynamic(numItems, capacity, matrix, items);

    //Store the knapsack results
    int numSacked = 0;
    int totalWeight = 0;
    int totalValue = 0;
    int jump = 0;
    item * knapSack = new item [numItems];
    for (int i = numItems; i > 0; i--)
    {
        if(matrix[i][capacity - jump] == matrix[i-1][capacity - jump])
            continue;
        knapSack[numSacked] = items[i-1];
        totalValue += knapSack[numSacked].value;
        totalWeight += knapSack[numSacked].weight;
        jump += knapSack[numSacked++].weight;
    }

    //Printing out the results
    cout << "Dynamic Programming Result:" << endl;
    cout << numSacked << endl;
    cout << totalWeight << endl;
    cout << totalValue << endl;
    for(int i = 0; i < numSacked; i++)
        std::cout << knapSack[i].name << " " << knapSack[i].value << " " <<
        knapSack[i].weight << std::endl;

    cout << endl;

    //Set the whole matrix to 0s to reset our matrix
    for (int i = 0; i < numItems + 1; i++)
```

```
69          for (int j = 0; j < capacity + 1; j++)
70              matrix[i][j] = 0;
71
72      //Refined Dynamic Programming - only calculate what you need
73      Refined_Dynamic(numItems, capacity, matrix, items);
74
75      //Find and Store the knapsack results
76      jump = 0;
77      numSacked = 0;
78      totalWeight = 0;
79      totalValue = 0;
80      item * TheSack = new item [numItems];
81      for (int i = numItems; i > 0; i--)
82      {
83          if(matrix[i][capacity - jump] == matrix[i-1][capacity - jump])
84              continue;
85          TheSack[numSacked] = items[i-1];
86          totalValue += TheSack[numSacked].value;
87          totalWeight += TheSack[numSacked].weight;
88          jump += TheSack[numSacked++].weight;
89      }
90
91      //Printing out the results
92      cout << "Refined Dynamic Programming Result:" << endl;
93      cout << numSacked << endl;
94      cout << totalWeight << endl;
95      cout << totalValue << endl;
96      for(int i = 0; i < numSacked; i++)
97          std::cout << TheSack[i].name << " " << TheSack[i].value << " " <<
            TheSack[i].weight << std::endl;
98
99      return 0;
100 }
```