A NIGHT IN A CRIME SCENE

Huge Ackman, a world known detective/secret agent, got a new case last night. He was informed that a crime was committed in the Downtown. The media and the news shared this tragedy as an incident but *Huge Ackman*, of course, knows that the Godfather *Alpha Chino* is behind it. His duty is to reach the crime scene and collect as much as valuable evidence in order to prove that the *Alpha* is guilty. But there is a problem. The crime scene and the evidence are heavily protected by the men of the mafia. The only way to reach the crime scene is to sneak in at night without alerting any security. For any other case, this would be an every-day-job for *Detective Huge*. However, this might be the last opportunity to bring down the *Alpha* after the years of chase.

Knowing that, the detective came to ***you***, a world class hacker working for the *FBI (Free Bogazici Intelligence)*, in order to get your help on this case. Being a genius, you saw an *if-condition* bug in the mafia's security system and you were able to reach the security cameras of the crime scene without even taking a sip of your coffee. Now, you are able to see all of the evidences that are waiting for *Huge* to collect. Thanks to your knowledge in advance machine learning, image recognition and *for-loops*, you ran an algorithm to detect various information about the evidences in the crime scene by looking from the security cameras. After a short coffee break, you got your results and listed the following instructions that the detective has to plan on before going in:

- Every evidence in the scene has an **id**, **weight**, and an **evidence value**. The more evidence value collected in total, the more chances that the mafia is proven to be guilty.
- Every evidence has also a **collection time**, representing the time required to collect the evidence without ruining it.
- You will be able to hack and distract the security system for a limited amount of time before the mafia notices something is happening. So the *Detective Huge* has **T** minutes after he steps into the crime scene.
- The detective will sneak in and start to fill his pockets. But his pockets have a limit of carrying maximum **W** weight. If he collects more than he can carry, the mafia will catch him on the run.

Your task is to write a program and help *the Detective Huge* to find the best subset of evidences that maximizes the total evidence value by considering the limitations above. Given the information you have, you are expected to output the total value and the list of items to be collected. The input file that gives the list of evidences is explained below:

Input Format(crime_scene.txt):

W T
N
$evidence\_id_1$ $weight_1$ $collection\_time_1$ $evidence\_value_1$
$evidence\_id_2$ $weight_2$ $collection\_time_2$ $evidence\_value_2$
…
$evidence\_id_N$ $weight_N$ $collection\_time_N$ $evidence\_value_N$

Input Format Explanation and Limitations:

In the first line of the input file 'crime_scene.txt' you will be given two integers respectively: **W** and **T**, which are the maximum weight and time limit. In the second line you will be given an integer number **N** which is the number of evidences. In the following N lines, you will be given 4 integers: the **evidence_id,** evidence **weight**, **collection_time**, and the **evidence_value** respectively.

Limitations

- (0 < **W** < 100)
- (0 < **T** < 100)
- (0 < **evidence_id** < 10000)
- (0 < **weight** < 100)
- (0 < **collection_time** < 100)
- (0 < **evidence_value** < 100)
- (0 < **N** < 10)
- The detective is so good at doing his job so he won't waste any time between collecting evidences. So assume T = 8 and there are 3 evidences with collection times 4, 3 and 1 respectively. The detective can collect all of the evidences within that time.
- Same rule above also applies to the maximum pocket capacity W. Assume W = 10 and there are 4 evidences with 5, 2, 1, 2 weight respectively. The detective can collect all of the evidences with that capacity.
- All evidence_id's are unique.
- Assume that there are no multiple subset of solutions that gives the maximum evidence value (Even if there are, you will get full points if you print out any of the subset of solutions)
- Assume that there will be at least one evidence that the detective can collect

Output Format:

total_value
evidence_id$_2$ evidence_id$_4$ evidence_id$_1$ ...

Output Format Explanation:

In the first line of the output, print the total value of your solution by summing up the evidence_value's of the collected evidences. On the second line, print the id's of the collected evidences. The order of the id's should be sorted from min to max. You should write your own sort function, any type of library function for sorting is forbidden. Second line of the output will be graded partially which will be explained below.

In order to make the project easier to implement, we are dividing it into 3 parts, and we give you partial points according to your output. For example, you will still get other partial points if your output is not sorted. You are expected to write a code that outputs 3 files for the given input which will be explained below.

Part 1: Solve the problem with only weights (35 Points)

Implement the project just by considering weights, maximum capacity W, and evidence values. In other words, assume that the detective has unlimited amounts of time. Find and print the best subset of items that gives the maximum evidence value to output file *'solution_part1.txt'*.

Partial Points:

You will get *20 points* if you print out the maximum total evidence value.
You will get 10 *points* if you also print out the evidence id list
You will get *5 points* if your id list is also sorted

Part 2: Solve the problem with only collection time (35 Points)

Implement the project just by considering collection time, the detective time limit T, and evidence values. In other words, assume that the detective has unlimited weight capacity. Find and print the best subset of items that gives the maximum evidence value to output file ***'solution_part2.txt'***.

Partial Points:

You will get *20 points* if you print out the maximum total evidence value.
You will get 10 *points* if you also print out the evidence id list
You will get *5 points* if your id list is also sorted

Part 3: Solve the problem with both weights and collection time (30 Points)

Implement the algorithm by considering weights, collection time, the detective time limit T, the weight capacity W, and evidence values. In other words, use all of the information given in the input file. Find and print the best subset of items that gives the maximum evidence value to output file ***'solution_part3.txt'***.

Partial Points:

You will get *15 points* if you print out the maximum total evidence value.
You will get 10 *points* if you also print out the evidence id list
You will get *5 points* if your id list is also sorted

Sample Input/Output and Explanations:

Example Input : (crime_scene.txt)

10 20
3
100 7 11 10
300 4 9 5
200 5 12 6

Input 1 Explanation

Maximum capacity W = 10 and time limit T = 20.
There are 3 items in the crime scene.
First one's id is 100, weight is 7, collection time is 11 and it has 10 evidence value.
Second one's id is 300, weight is 4, collection time is 9 and it has 5 evidence value.
Third one's id is 200, weight is 5, collection time is 12 and it has 6 evidence value.

Solution of the Input 1 with only weights (Part 1):

The detective has 10 weight capacity. The best option would be collecting evidence 300 and evidence 200. Because the weights of those two are 4 + 5 = 9 which doesn't exceed the capacity W=10, and they add up to 5 + 6 = 11 evidence value in total. Notice that the detective could collect the evidence 100 with 7 weights and 10 evidence value, but if he does that, he can't collect any other evidences because it would exceed the 10 weight limit (7 + 4 > W or 7 + 5 > W) so it is not the optimal solution.

The maximum evidence value is 11 and the best subset of items are [300,200]. So, the output ***solution_part1.txt*** will be:
11
200 300

Solution of the Input 1 with only collection time (Part 2):

The detective has a 20 time limit. The best option would be collecting evidence 100 and evidence 300. Collection time of these two are 11 + 9 = 20 which does not exceed the time limit, and the total evidence value is 10 + 5 = 15.
The maximum evidence value is 15 and the best subset of items are [100,300]. So, the output ***solution_part2.txt*** will be:
15
100 300

Solution of the Input 1 with both weights and collection time (Part 3):

The detective has 10 weight capacity and also 20 time limits. If he collects evidence 100 and 300, the time limit is not exceeded (11+9=20), but the weight capacity is exceeded (7+4=11). Similarly if he collects evidence 300 and 200, the weight capacity is not exceeded (4+5=9), but the time limit is exceeded (9+12 = 21). So he only collects evidence 100 which is the most valuable single piece.
output ***solution_part3.txt*** will be:
10
100

Example Input 2 : (crime_scene.txt)

10 20
6
100 7 11 10
200 4 9 5
300 5 12 6
150 1 3 8
133 7 5 9
111 3 1 3

Part1 Output : (solution_part1.txt)

19
150 200 300

Part2 Output : (solution_part2.txt)

30
100 111 133 150

Part3 Output : (solution_part3.txt)

18
100 150

Restrictions and Hints

- This project is prepared to be solved specifically with recursive functions. Do not try to find every possible combination with for-loops. You can use for loops while reading/writing the data, or sorting the output list. BUT other than those parts, **it is forbidden to use for loops** inside of your solution.
- Please write your own sort function as described in the project. Do not use any library function for sorting.

Hint 1: Even if using for-loops is restricted, recursive functions can easily replicate them and help you to iterate over a list. Please inspect the following code and understand what it does:

```
def f(N, i):
    if i == N:
        return 0, []
    f_sum, f_list = f(N, i+1)
    f_sum += i
    f_list.append(i)
    return f_sum, f_list
```

Hint 2: Think about a recursive function that gives you the maximum evidence value with the available weights and times when you are in front of an evidence. Start from the first item. You have two options: take it if your weight and time resources allow it, or leave it and don't waste your resources on that evidence (there may be better options in the next evidences). If you take it, you will have the evidence value of the item in your hand, it will add up to your weight and time limit, and you will try to calculate the maximum evidence value with the remaining resources on the next items. If you don't take it, you won't waste any resources on this evidence, and you will try to calculate the maximum evidence value on the next items with the resources you didn't waste. Select whichever option gives you the bigger total evidence value. That will be the optimal output for the evidence you are standing in front of.

Please submit your code in a single file named '_<student_number>.py'