BARICA

Barica is an unusual frog. She lives in a pond where N plants float on the surface of the water. The plants are numbered 1 through N. When viewing from above, the location of each plant is given by a pair of coordinates. What makes Barica unusual is her fear of jumping diagonally and in the negative direction. More precisely, she can jump from a plant at coordinates (x1, y1) to another at coordinates (x2, y2) only if:

- x2 > x1 and y2 = y1, or
- y2 > y1 and x2 = x1

For each plant, we know the number of flies in its immediate vicinity. Barica can use her swift tongue to eat all flies near the plant she is at.

Barica absorbs one energy unit for each fly she eats, and uses K energy units for each jump she makes. Barica can not make a jump if she doesn't have enough energy units beforehand.

Barica wants to go from plant 1 to plant N and have the **largest amount of energy** possible after arriving. Barica initially has no energy and must gather energy for her first jump from the flies around plant 1.

Find the sequence of plants Barica should travel to achieve her goal.

Input

The first line of input contains two integers N and K (2 ≤ N ≤ 300 000, 1 ≤ K ≤ 1000) separated by a space.

Each of the following N lines contains three integers X, Y and F (0 ≤ X, Y ≤ 100 000, 0 ≤ F ≤ 1000) separated by spaces, meaning that there is a plant at coordinates (X, Y) with F flies around it.

The first plant in the input is plant 1, the second is plant 2 etc.

No two plants will share the same pair of coordinates.

Note: The input data will guarantee that a sequence of jumps, although not necessarily unique, will always exist.

Output

Output the final energy level on the first line.

Output an integer L, the number of plants Barica should travel, including plants 1 and N.

On the following L lines, output the sequence of plants Barica should travel.

Example

Input:

Input:

Input:

```
6 5
1 1 5
2 1 5
1 2 4
2 3 5
3 2 30
3 3 5
```

Output:

```
5
4
1 1
2 1
2 3
3 3
```

```
8 10
1 1 15
2 2 30
1 2 8
2 1 7
3 2 8
2 3 7
4 2 100
3 3 15
```

Output:

```
36
5
1 1
1 2
2 2
3 2
3 3
```

```
9 5
5 5 10
6 5 2
7 5 1
5 6 2
6 6 6
7 6 2
5 7 1
6 7 2
7 7 1
```

Output:

```
2
3
5 5
7 5
7 7
```