Broken Tetris

As a kid, Lea loved playing Tetris on her Gameboy. Since then, her favorite console had to endure a lot, from being thrown ragefully into various corners to tears of joy from beating the highscore. Unfortunately, this left a lot of marks such that some features of the game are no longer available:

- Pieces can no longer be rotated.
- Pieces can no longer be moved.
- Pieces spawn at random horizontal positions.
- Only rectangular pieces can spawn.
- The size of the playing field varies from game to game.
- Full rows are no longer erased.

Since Tetris was her favorite game growing up, she cannot help but turn the game on every once in a while and just watch the pieces fall down. As the playing field can get so huge, that it is no longer clear which point of the field is the highest, Lea wants you to write a program that shows the height of the highest position on the playing field at any point during the game. Can you help her out?

Input

The first line of the input contains an integer t. t test cases follow, each of them separated by a blank line.

Each test case begins with two integers n and k. n is the width of the playing field, k is the total number of blocks to spawn. k lines follow describing the blocks. Each line contains three integers w, h and p. w and h give the width and height of the block, p the offset from the left side of the playing field, i.e. $p = 0$ means that the block spawns at the leftmost position of the playing field.

Output

For each test case output one line containing "Case #i: $r_1 \ldots r_k$" where i is its number, starting at 1, and r_j is the maximum height of the stacked blocks on the playing field after the j-th block has landed.

Constraints

- $1 \leq t \leq 5$
- $2 \leq n \leq 1000000$
- $1 \leq k \leq 100000$
- $1 \leq w, h \leq 100$
- $0 \leq p < p + w \leq n$

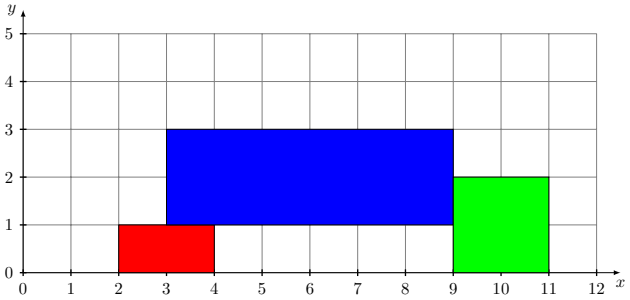

Figure 1: Visualization of first sample input

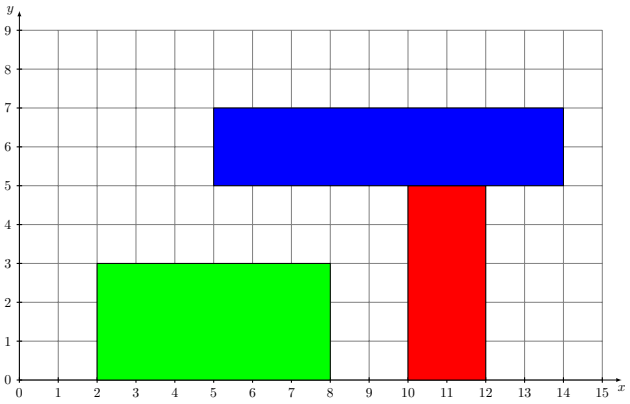

Figure 2: Visualization of second sample input

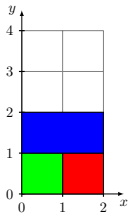

Figure 3: Visualization of third sample input

Sample Input 1

```
3
12 3
2 1 2
6 2 3
2 2 9

15 4
2 5 10
6 3 2
9 2 5
1 1 1

2 3
1 1 1
1 1 0
2 2 0
```

Sample Output 1

```
Case #1: 1 3 3
Case #2: 5 5 7 7
Case #3: 1 1 3
```

Sample Input 2

```
4
7 9
3 4 2
4 4 3
4 2 1
1 2 4
4 4 2
3 3 2
5 2 1
1 3 2
3 1 1

13 10
3 2 3
3 2 4
4 5 4
1 5 6
4 2 0
2 5 4
2 2 8
5 2 7
4 3 5
5 5 3

12 3
4 3 8
3 4 6
2 3 9

13 9
3 4 4
1 5 3
4 2 2
2 1 7
3 4 4
5 4 4
4 3 5
1 3 11
1 2 0
```

Sample Output 2

```
Case #1: 4 8 10 12 16 19 21 24 25
Case #2: 2 4 9 14 14 14 14 14 17 22
Case #3: 3 7 7
Case #4: 4 5 7 7 11 15 18 18 18
```