Orders

You have just been hired as the delivery manager of a restaurant. As a delivery manager, you want to manage the orders that arrive and process them accordingly. Your restaurant can process multiple orders simultaneously without any problems and you are interested in knowing when will each order be finished.

To ease the process, you are going to create an order management system that can take multiple orders and process them according to their time of arrival. After all, you do not want your customers to be angry with you if you do not do so.

Each order has its own order number, which starts from 1 and assigned according to the order of arrival of the orders. The restaurant can handle up to K orders at the same time (since it has K processors). The processors are numbered from 1 to K and any new order must be assigned to the lowest-numbered available processor. If there are no processors currently available, this order will just wait for the next readily available processor.

Your job is to simulate this order management system and print when will each order be finished and by which processor. Since this is not an easy task to do and involves a lot of corner cases, you must be careful in implementing this order management system so it does not mess up the orders. You do not want to get into trouble for mixing things up, do you?

Good Luck!

Input

The first line of input consists of two integers K (1 <= K <= 10) and O (1 <= O <= 10000), the number of processors the restaurant has and the number of orders on that particular day, separated by a single space. After that, Q rows follow. Each of the Q rows contains two integers T and X. T refers to the time of arrival of that order (in minutes after the restaurant opens) and X refers to the time (in minutes) needed to process this particular order. It is guaranteed that the order will come according to its time of arrival and starts with order number 1 up until order number O.

Output

For each order, print its order number, the number of the processor that processed that particular order, and the time (in minutes after the restaurant opens) that the order is processed, all in a single line separated by a single space. You must print the list of orders sorted by its order number, starting from order number 1 until the last order.

Sample Input 1

```
1 3
0 9
5 5
10 1
```

Sample Output 1

```
1 1 9
2 1 14
3 1 15
```

Sample Input 2

```
2 3
0 9
5 5
10 1
```

Sample Output 2

```
1 1 9
2 2 10
3 1 11
```

Explanation

In the first sample input, there is only one processor. There are three orders. Order number one comes at the time the restaurant opens (0th minute) and will be finished in the 9th minute. Order number two comes at the fifth minute, but then there is only one processor and it is currently busy processing another order. It will be processed starting from the start of the 10th minute and will be finished at the end of the 14th minute, needing 5 minutes to complete. The last order will be processed in the 15th minute and will be finished in the 15th minute as well, only needing 1 minute to be completed. All orders are processed by processor number 1.

In the second sample input, we now have two processors. Everything else is the same. Now, the second order can be processed at the time of its arrival, and be completed at the 10th minute. When order number three comes, there are two free processors (1 and 2) since processor 2 has just finished processing the second order. But we need to assign this to processor number 1 since it is a lower-numbered processor. Hence, processor number 1 will process this order and it will be finished by the 11th minute.

Skeleton

You are given the skeleton file `Orders.java`. You should see a non-empty file when you open the skeleton file. Otherwise, you might be in the wrong working directory.

```
/**
Name        :
Matric No.  :
PLab Acc.   :
*/

import java.util.*;

public class Orders {
    public void run() {
        // implement your "main" method here
    }

    public static void main(String[] args) {
        Orders o = new Orders();
        o.run();
    }
}
```

Notes:

1. You should develop your program in the subdirectory **`ex1`** and use the skeleton java file provided. You should not create a new file or rename the file provided.
2. If your algorithm is different from the given skeleton, you are free to write a solution according to your own algorithm. You must use **either Stack and/or Queue** in your main algorithm to solve this problem. <Restriction on Arrays/etc. but they need to use array somehow>.
3. You are **free to define your own classes (or remove existing ones)** if it is suitable.
4. Please be reminded that the marking scheme is:

   Input : 10%
   Output : 10%
   Correctness : 50%
   Programming Style : 30% (awarded if you score **at least 20% from the above**):
   - Meaningful comments (pre- and post- conditions, comments inside the code): 10%
   - Modularity (modular programming, proper modifiers [public / private]): 10%
   - Proper Indentation: 5%
   - Meaningful Identifiers (for both method and variable names): 5%

   Compilation Error : Deduction of **50% of the total marks obtained**.