Computer Based Test

1. *For each question, there will be a testcase (e.g. the testcase for question 1 is* testcase1.txt) *Please note that the testcase you received may not necessarily to be the same as the testcase used for grading. Please consider edge cases.*

2. *Let your program read the corresponding testcase and print the result to a corresponding text file(e.g. the output for question 2 should be* output2.txt*).*

3. *For each question, we have provided a corresponding answer (e.g assume you're solving problem 3, the corresponding answer is* ans3.txt*), and your output should be same as answer.*

4. *For each question, your program file should be named* DS{problem number}_{student ID}.c. *(e.g. assume your student ID is* 7112056067 *and you're solving question 3, your program file should be* DS3_7112056067.c)

5. *Each question is scored independently, but partial credit is not awarded; full credit is given only for complete correctness.*

6. *You can submit for three times.*

1 Problem 1

Write a program to simulate the process scheduling of an operating system. Assume that the system has only one CPU, and each process has a known **arrival time**, **execution time**, and **priority level**. The priority level is represented by a natural number, where a **higher number indicates a higher priority**.

Rules

1. If a process arrives when the CPU is idle, it will **occupy the CPU until it completes**, unless a new process with a **higher priority** arrives. In this case, the new (higher-priority) process will **preempt** the CPU, and the previous process must wait.
2. If a process arrives while the CPU is executing a process with a **higher or equal priority**, the newly arrived process must **wait**.
3. Once the CPU becomes idle, if there are waiting processes, the process with the **highest priority** will be executed first.
    - If multiple processes have the same highest priority, the process that arrived **earlier** will be selected.

Input Format

The input consists of multiple lines, the first line represents the number of test cases. Begins with number of task(s), and each case containing four natural numbers (each not exceeding 10^8), representing:

- **Process ID**
- **Arrival Time**
- **Execution Time**
- **Priority Level**

Each process has a unique process ID. No two processes with the same priority level will arrive at the same time. The input is **sorted in ascending order** by arrival time. The waiting queue will never exceed **15,000** processes at any time.

Output Format

For each process, output its **process ID** and **completion time** in the order they finish execution.

Sample Input

```
2
5
1 1 50 1
```

```
2 10 40 2
3 20 30 3
4 30 20 4
5 40 10 1
8
1 1 5 3
2 10 5 1
3 12 7 2
4 20 2 3
5 21 9 4
6 22 2 4
7 23 5 2
8 24 2 4
```

Sample Output

```
4 50
3 70
2 100
1 141
5 151

1 6
3 19
5 30
6 32
8 34
4 35
7 40
2 42
```

2 Problem 2

Given an undirected graph, find all the edges that cannot be part of **any** minimum spanning tree (MST). These edges are called **heavy cycle edges**, meaning that they would always create a cycle with a higher total weight and are therefore never selected in any MST.

Input

The input consists of several datasets. Each dataset begins with a line containing two integers:

- n – the number of nodes ($0 < n \leq 1000$),
- m – the number of edges.

Each of the next m lines contains three integers:

- u, v – two endpoints of an edge,
- w – the weight of the edge connecting u and v.

The input terminates with a line containing `0 0`.

Output

For each dataset, print a sorted list (in ascending order) of the weights of the edges that **will not appear in any minimum spanning tree**. If there are no such edges, print:

```
forest
```

Note

All edge weights are non-negative. The graph is connected in each test case.

Sample Input

```
3 3
0 1 1
1 2 2
2 0 3
4 5
0 1 1
1 2 2
2 3 3
3 1 4
0 2 0
3 1
0 1 1
0 0
```

Sample Output

```
3
2 4
forest
```

3 Problem 3

In a simplified computer system, memory pages are managed using an LRU (Least Recently Used) cache mechanism. Initially, the memory contains n pages labeled from 1 to n, ordered from the most recently used (on top) to the least recently used (at the bottom).

Rules

- First line is the number of test cases, at most 100.
- One line with two integers n and m $(1 \leq n, m \leq 100000)$ — the initial number of pages and the number of access requests.
- One line with m integers $a_1, a_2, \ldots, a_m$ $(1 \leq a_i \leq n)$ — the sequence of page access requests.
- Initially, pages are ordered from 1 (top) to n (bottom), where page 1 is the most recently used.
- For each test case, print a single line with m integers. The i-th integer should represent the number of pages that were more recently used than page a_i *before* it was accessed and moved to the top.
- The time complexity must be in $\mathcal{O}(\log N)$.(Segment tree)

Sample Input

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

Sample Output

```
2 1 0
3 0 4
```