05. Operation Data Breach

Condition:

A cyberattack by an unknown hacking group has compromised a database containing classified information. Each record in the database is an encrypted fragment of data stored in a hierarchical structure called **a "Cryptographic Tree"** .

A cryptographic tree is similar to a standard tree, but with specific rules:

1. Each node contains an encrypted record.
2. Each child depends on its parent via a decryption key that is derived based on the state of the parent.
3. The root of the tree is the only node that can be decrypted directly with the "Master Key".

The hacking group has managed to obtain the "Master Key" and the cryptographic tree, but to extract all the information they need an automated decryption process. Your task is to help the cybersecurity team restore the compromised database by decrypting all the nodes in the tree.

Decryption rules:

1. Each node has a unique identifier **ID** and an element **E** , which contains the encrypted record.
2. Decrypting element **E** requires the following steps:
    - If the node is the root (has no parent), use the "Master Key".
    - If the node has a parent, generate a key from the decrypted element of the parent by hashing with SHA256.
    - Decrypt the node element using the resulting key.
3. The tree data is presented in a hierarchical format, with each node described by:
    - **ID** (unique identifier).
    - **ParentID** (parent node identifier; if -1 , the node is the root).
    - **E** (the encrypted entry).
4. After decrypting a node, the resulting content should be saved to the node for further use.

Input:

1. The first line gives **N** ($1 \le$ **N** $\le 10{,}000$) – the number of nodes in the tree.
2. The following are **N** lines, each containing:
    - **ID** ($1 \le$ **ID** $\le 10{,}000$).
    - **ParentID** (or -1 if root).
    - **E** (encrypted record as string).
3. The last line contains **the MasterKey** – the master key for decrypting the root node.

Output:

Output the decrypted information for all nodes, sorted by their IDs.

Example:

Input	Output
5 1 -1 "f1e2d3c4" 2 1 "a1b2c3d4" 3 1 "e4d3c2b1" 4 2 "abc123ef" 5 3 "efc1b2a3" "key_master_001"	1: "data_root" 2: "data_child1" 3: "data_child2" 4: "data_leaf1" 5: "data_leaf2"

Additional conditions:

- It is guaranteed that the input will always represent a valid tree.
- SHA256 hashing should be performed on strings, without additional formatting.
- Decryption of elements can be emulated using the decrypt(E, Key) function (a secure library function).
- Your solution must be efficient to handle large input data.

Solving instructions:

1. **Building the tree:**
    - Organize nodes in a hierarchical structure using ParentID .
2. **Decryption:**
    - Start from the root, decrypt its element with the master key.
    - Recursively traverse the children, generating keys via SHA256 from the already decrypted elements of the parents.
3. **Recursion or BFS:**
    - Use recursion or breadth-first traversal to process all nodes.
4. **Efficiency:**
    - Organize the data in a format to quickly find the children of each node (e.g. hash tables).