### **Simplified Question: Chunk-Based File Assembly**

You are tasked with building a simple **client-server system** where:

1. Clients send files to a server in **chunks**, and the server must **reconstruct the file** in the correct order.
2. Chunks can arrive **out of order**, and the server must reorder them before combining.

### **Constraints:**

1. Each chunk is labeled with a unique **file ID** and a **sequence number** (chunk\_id).
2. The system supports multiple files from different clients.
3. Assume all chunks for a file are eventually received.

### **Objective:**

Write Python logic to process a list of chunks at the server:

1. Reorder chunks using their chunk\_id.
2. Print the **reconstructed file** for each file\_id.

### **Input Format:**

A list of tuples, where each tuple represents a chunk:

python

Copy code

chunks = [(file\_id, chunk\_id, data), ...]

* file\_id: The file to which the chunk belongs (string).
* chunk\_id: The sequence number of the chunk (integer).
* data: The content of the chunk (string).

### **Output Format:**

For each file, print the reconstructed file in the correct order:

python

Copy code

File <file\_id>: <reconstructed\_file\_data>

### **Example Input:**

python

Copy code

chunks = [  
 ("file1", 3, "C"),  
 ("file2", 1, "X"),  
 ("file1", 1, "A"),  
 ("file1", 2, "B"),  
 ("file2", 2, "Y"),  
 ("file1", 4, "D")  
]

### **Example Output:**

plaintext

Copy code

File file1: ABCD  
File file2: XY

### **Hint:**

1. Use a **dictionary** to group chunks by file\_id.
2. Use Python's sorted() to reorder the chunks by chunk\_id.
3. Concatenate the data of chunks to reconstruct the file.

### **Bonus Challenge:**

1. Modify your solution to handle missing chunks by printing a warning if chunks are skipped (e.g., missing chunk IDs).