Problem 144: Little Orphan ASCII

Difficulty: Medium

Authors: Cindy Gibson and Danny Lin, Greenville, South Carolina, United States

Originally Published: Code Quest 2021

Problem Background

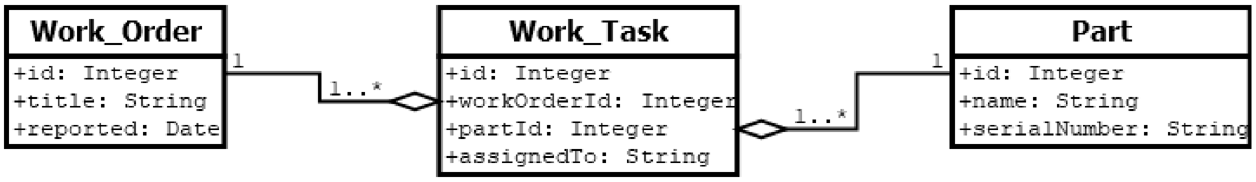
Databases are a critical part of many software systems. A database is sort of like a digital filing cabinet - it's a large repository for any sort of information your software program might need. As the program runs, it can add data to the database, modify it, or delete it. Many times, data records in your database will refer to other data records. For example, a record about a library book may contain a reference to the library from which it was checked out.

Maintaining these relationships between different parts of your data is called "data integrity." Most databases automatically manage data integrity - to use the example above, it doesn't make sense to say a book belongs to a library that doesn't exist. However, if a database isn't set up correctly, these relationships can break down, causing "orphaned data." Perhaps the library did exist once, but has since closed and thus been removed from the database. Unless their records are updated or deleted, any books in the library are now orphaned. We know they exist, but we have no idea where they belong. This can cause problems for a software application that expects every book to be neatly stored somewhere on a library shelf.

Problem Description

The US Army has contracted Lockheed Martin's Rotary and Missions Systems division to update its maintenance tracking software. Their existing system has become badly outdated, and wasn't set up with proper data integrity protections. Your task is to go through the old system's database and identify any records that have become orphaned so they can be reported to the Army and investigated.

The main area of concern is three tables in the database, illustrated below:



The Work_Order table contains information about problems that have been reported on the Army's equipment. Each work order contains one or more Work_Tasks, which detail the specific jobs that need to be completed to fix the reported problem. Each work task requires a Part from the supply in order to be completed. These relationships can be seen by the lines extending from the Work_Task table; the "workOrderId" field should contain a value matching the "id" value of an entry in the Work_Order table, and "partId" should refer to a Part's "id" value.

Unfortunately, when the database was set up, these relationships were not clearly defined. As a result, over time the database has become corrupted, leaving orphaned task data that refers to a non-existent work order, a non-existent part, or both. Your task is to write a program that scans the database to identify these orphaned records, so the Army can search through their physical paperwork to rebuild their maintenance records.

Sample Input

The first line of your program's input, received from the standard input channel, will contain a positive integer representing the number of test cases. Each test case will include:

- A line containing three positive integers, separated by spaces:
  - W, representing the number of work orders in the database
  - T, representing the number of work tasks in the database
  - P, representing the number of parts in the database
- W lines representing the data stored in the Work_Order table. Each line contains the following information, separated by commas (,):
  - A non-negative integer representing the ID value of the Work_Order. Each ID value will be unique amongst other Work_Order entries.
  - A text string (containing upper- and lower-case letters, spaces, and/or numbers) representing the title of the Work_Order
  - A date, in DD-MM-YYYY format, representing the date on which the Work_Order was reported.
- T lines representing the data stored in the Work_Task table. Each line contains the following information, separated by commas (,):
  - A non-negative integer representing the ID value of the Work_Task. Each ID value will be unique amongst other Work_Task entries.
  - A non-negative integer representing the ID value of the associated Work_Order. This number may not appear in the provided Work_Order entries.
  - A non-negative integer representing the ID value of the associated Part. This number may not appear in the provided Part entries.
  - A text string (containing upper- and lower-case letters and/or spaces) representing the name of the technician assigned to this Work_Task.
- P lines representing the data stored in the Part table. Each line contains the following information, separated by commas (,):
  - A non-negative integer representing the ID value of the Part. Each ID value will be unique amongst other Part entries.
  - A text string (containing upper- and lower-case letters, spaces, and/or numbers) representing the name of the Part.

  - A text string (containing upper- and lower-case letters, dashes, and/or numbers) representing the serial number of the Part.

```
1
3 4 3
1,Dead Battery,29-04-2020
3,Flat Tire,01-05-2020
5,Door Jammed,02-05-2020
1,1,2,John Doe
2,2,4,Jane Doe
3,3,5,James Doe
4,4,7,Joan Doe
2,Truck Axle,12-34
4,Spare Tire,7890
6,Thingiemabob,3-14159
```

Sample Output

For each test case, your program must print information about each Work_Task entry that contains orphaned data. Entries with orphaned data should be printed in increasing numerical order by their ID value. Each line should include the following information, separated by spaces:

- The ID number of the orphaned Work_Task entry
- If the Work_Order ID is invalid, the phrase "MISSING WORK_ORDER ##", replacing ## with the invalid Work_Order ID
- If the Part ID is invalid, the phrase "MISSING PART ##", replacing ## with the invalid Part ID

```
2 MISSING WORK_ORDER 2
3 MISSING PART 5
4 MISSING WORK_ORDER 4 MISSING PART 7
```