Assignment 4B: JDBC TakeHome

1: This assignment can be done in a group of 2 students; individual submissions are fine, but won't get extra credit!
2. This is a take home assignment. You may work on it in the lab if you finish 4A.

Part 1: Create a Java class ResultSetOutput with two static methods toHTML() and toJSON() which take as argument a JDBC ResultSet, and prinout out the result in HTML and JSON format respectively, using ResultSet metadata functions.

For HTML, the result should be printed as a table, with a header containing column names, followed by the data. E.g.

```
<table>
   <tr><th>ID</th> <th>name</th></tr>
   <tr> <td>1234</td> <td>John</td> </tr>
   <tr> <td>6789</td> <td>Shyam</td> </tr>
</table>
```

For JSON, the result should be a JSON object with a header: attribute which is an array of column names, and a data: attribute, which is an array of objects, one per row, where the object for a row has attribute:value pairs for each attribute of the row.

```
E.g.  {"header": ["ID", "name"],
        "data": [ {"ID:"1234", "name":"John"},
                   {"ID":"6789", "name":"Shyam"}
                 ]
       }
```

Write a function that takes an SQL query as input from the terminal (in a single line) and executes the query and prints the resultset in the HTML and JSON formats.

What to submit: You will submit a Java file 4B_*Part1_<roll_no1>_<roll_no2>.java* with a main() that does the above.

Part 2: Parts-Explosion

Consider a device that has many parts, with the root part called root, where each part has a local cost, and may have subparts, with a subpart possibly occurring multiple times in a part. This information is represented by the following relations.

part(ID, localcost)

subpart(pID, spID, number)

where number indicates how many times spID occurs in pID. Here is some sample data

part('root', 100), part('wheel', 50), part('tyre', 20), part('engine', 200),

subpart('root', 'engine', 1), subpart('root', 'wheel', 4), subpart('wheel', 'tyre', 1)

The cost of a part is computed from its local cost and the cost of its subparts (which in turn may depend on the cost of their subparts, and so on). In the above example, the cost of wheel is 70, and of root is 580.

To compute the cost, you should perform an aggregation in JDBC code, based on a topological sort of subpart.

You must do the topological ordering using SQL, by a recursive query of the following form:

heights(ID, 1) if part(ID, …)

heights(ID, i+1) if subpart(ID, ID1, n) and heights(ID1, i)

and define maxheight(ID, height) is the maximum of heights for that ID.

NOTE: to avoid infinite recursion in case there is a cycle in the input, you should add a condition that height < 100 (or any other limit) in the recursive definition. Such cycles should not occur in a correct subpart hierarchy, this is just for error handling.

Your JDBC program should create a temporary relation cost(ID, cost), which is initially empty; executeUpdate() can be passed DDL commands, and to create a temporary table use the syntax **create temporary table ...** . Note that two concurrent executions of the program will see their own tables, and will not conflict.

It should then consider part IDs in increasing order of their maxheight, and compute the cost based on the local cost plus the already computed costs of the subparts (if any), and save the computed cost of ID in the cost relation.

At the end it should look up the cost of the part ID taken from the terminal and print it out.

The input should be part IDs entered one per line from the terminal (have a loop so we can test on multiple IDs), and the output should be of the form:

```
ID   cost
```

with each ID in a line by itself. Terminate on end of file (ctrl-D from terminal/console).

What to submit: You will submit a Java file 4B_*Part2_<roll_no1>_<roll_no2>.java* with a main() that does the above.

Table definitions and sample data:

```
create table part(

    ID varchar(20) primary key,

    cost int);

create table subpart (

    pID varchar(20) references part,

    spID varchar(20) references part,

    number int);

insert into part values ('root', 100), ('wheel', 50), ('engine', 200), ('tyre',
20);

insert into subpart values ('root', 'engine', 1), ('root', 'wheel', 4),
('wheel', 'tyre', 1);
```