Normalization

INSTRUCTIONS

1. This project is an individual project.
2. Submit only one ZIP file, `<student number>.zip` (for example: "`A012345L.zip`"), containing your report (explanation of algorithm for Question 2b and 2c) (PDF file) and the completed python file "`project2.py`" (follow the template indicated in the file and feel free to add other necessary functions) to the folder "Normalization Submissions" in Luminus Files by **Sunday 3 March midnight**.
3. After the deadline and until Friday 8 March at 18:30, you can submit to the folder "Normalization Late Submissions" in Luminus Workbin (Files) (penalties apply).
4. Make sure to write your **name** and your **student number** on the front page of the report.
5. You may be asked to demonstrate the code that you submitted. Make sure that your code is readable and commented properly. Highlight the major algorithmic design choices in your comments. Do not forget to put your student number in the code.
6. Do not copy any code or part of the code from a third party (e.g. friend, internet source, book) as it will be considered plagiarism. If you borrow ideas from a source, add the necessary reference in the comments of your code and in your report, as appropriate.

In this project, we use Python 2.7. Use the Python file "`project2.py`" from Luminus files to answer to questions.

For all questions, we use the following representations. The schema is represented as a set of attributes between quotes. For example, the schema $\{A, B, C, D\}$ is represented as the Python list `['A', 'B', 'C', 'D']`. A functional dependency is represented as a list of pairs of list. For example, the functional dependency $\{AB\} \rightarrow \{C\}$ is represented as the Python term `[['A', 'B'], ['C']]`. A set of functional dependencies is a list of functional dependencies.

You can assume that the input to the questions is always valid. You do not need to validate the input. The code should be readable and commented properly.

Question 1 [6 marks]

(a) Write a function `closure` that takes three parameters, a schema, a set of functional dependencies and a set of attributes, subset of the schema, and returns the closure of the set of attributes. (3)

For example, the attribute closure of $\{A\}$, $\{A\}^+ = \{A\}$,

```
closure(['A', 'B', 'C', 'D'], [[['A', 'B'], ['C']], [['C'], ['D']]], ['A'])
      = ['A']
```

For example, the attribute closure of $\{A, B\}$, $\{A, B\}^+ = \{A, B, C, D\}$,

```
closure(['A', 'B', 'C', 'D'], [[['A', 'B'], ['C']], [['C'], ['D']]], ['A', 'B'])
      = ['A', 'B', 'C', 'D']
```

(b) Write a function `all_closures` that takes two parameters, a schema and a set of functional dependencies, and returns the closure of all subsets of attributes excluding super keys that are not candidate keys. The set of results is represented as a list of pairs of lists. (3)

For example,

```
all_closures(['A', 'B', 'C', 'D'], [[['A', 'B'], ['C']], [['C'], ['D']]])
      = [[['A'], ['A']], [['B'], ['B']], [['C'], ['C', 'D']], [['D'], ['D']],
      [['A', 'B'], ['A', 'B', 'C', 'D']], [['A', 'C'], ['A', 'C', 'D']],
      [['A', 'D'], ['A', 'D']], [['B', 'C'], ['B', 'C', 'D']],
      [['B', 'D'], ['B', 'D']], [['C', 'D'], ['C', 'D']], [['A', 'C', 'D'],
      ['A', 'C', 'D']], [['B', 'C', 'D'], ['B', 'C', 'D']]]
```

Question 2 [14 marks]

(a) Write a function `min_cover` that takes two parameters, a schema and a set of functional dependencies, and returns a minimal cover of the functional dependencies. The set of results is represented as a list of pairs of lists. (4)

For example,

```
min_cover(['A', 'B', 'C', 'D', 'E', 'F'],
         [[['A'], ['B', 'C']],[['B'], ['C','D']], [['D'], ['B']],
          [['A','B','E'], ['F']]])
      = [[['A'], ['B']], [['B'], ['C']], [['B'], ['D']],
         [['D'], ['B']], [['A', 'E'], ['F']]]
```

There might be more than one correct answer to this question.

(b) Write a function `min_covers` that takes two parameters, a schema and a set of functional dependencies, and returns all minimal covers reachable from the set of functional dependencies. The set of results is represented as a list of pairs of lists. Your report consists of up to one page explanation of the algorithm you devise to compute all minimal covers. (5)

For example,

```
min_covers(['A', 'B', 'C'], [[['A', 'B'], ['C']],[['A'], ['B']], [['B'], ['A']]])
      = [[[['A'], ['C']], [['A'], ['B']], [['B'], ['A']]],
         [[['B'], ['C']], [['A'], ['B']], [['B'], ['A']]]
        ]
```

(c) Write a function `all_min_covers` that takes two parameters, a schema and a set of functional dependencies, and returns all minimal covers. The set of results is represented as a list of pairs of lists. Your report consists of up to one page explanation of the algorithm you devise to compute all minimal covers. (5)

– END OF PAPER –