Project: Dependencies

Instructions

- This project is an individual project.
- The maximum mark for this project is 10 points for 10% of the final mark.
- Submit your answers by **Friday 25 February 2022, 17:00** to Luminus.
- There is strictly no late submission.
- Download the template Python file "`answers.py`" from Luminus
  "`Files > Projects > Dependencies`".
- Follow the requirements and modify the Python file as instructed. Feel free to add the necessary ancillary functions.
- Make sure that your code is readable and commented properly. Highlight the major algorithmic design choices in your comments.
- Submit the completed Python file "`answers.py`" to Luminus
  "`Files > Projects > Dependencies > Submissions`".
- There is no late submission.
- You may be asked to demonstrate the code that you submit.
- Do not copy any code or part of the code from a third party (e.g. friend, internet source, book) as it is plagiarism. If you borrow ideas from a source, add the necessary references in the comments of your code.

In this project, we use Python 3 to implement several algorithms related to functional dependencies that you have learnt in the lectures and tutorials.

Use the template Python file "`answers.py`" to answer the questions. Do not change the name and the parameters of the functions in the template file. You may add ancillary functions, if necessary or convenient. You may import additional Python standard libraries (e.g `itertools`, `copy`) but no external libraries.

For all questions, we use the following data structures. The schema is represented as a Python list of non-empty strings. For example, the schema $\{A, B, C, D\}$ is represented as `['A', 'B', 'C', 'D']`. Even though it is represented as a list, the order of the elements in the list does not matter. The list cannot contain duplicates. A functional dependency is represented as a list of two lists. The order of the two elements matters. For example, the functional dependency $\{A, B\} \rightarrow \{C\}$ is represented as `[['A', 'B'], ['C']]`. A set of functional dependencies is a list of functional dependencies. The order of the functional dependencies does not matters.

Ensure that your functions process the input and produce the output in the format specified in the corresponding questions. Functions that do not process the input and produce the output in the specified format shall receive zero mark.

You may assume that the input is always valid (e.g. no empty string, no incomplete functional dependency). Do consider the case in which the left- or right-hand side of a functional dependency is an empty set. The code should be readable and adequately commented.

Change the result of the function `student_number` in "`answers.py`" to your student number.

1. (3 points) (a) Write a function `closure` that takes three parameters: a schema, a set of functional dependencies and a set of attributes, which is the subset of the schema and returns the closure of the set of attributes.

   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 results is represented as a list of functional dependencies of the form $S \rightarrow S^+$ where S is a subset of attributes.

   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']]]
   ```

2. (7 points) (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 results is represented as a list of functional dependencies.

   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 results is represented as a list of minimal covers. Make sure that you clearly explained the rationale of the algorithm for this question in the indicated comment section in the code.

   For example,

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

   In this example, the function `min_covers` returns a list containing only one set of functional dependencies since it finds only one minimal cover.

(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 results is represented as a list of minimal covers. Make sure that you clearly explain the rationale of the algorithm for this question in the indicated comment section in the code.

For example,

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

For the case above, the function `all_min_covers` finds five minimal covers.