Automated Software Engineering Exam 2 (January 16, 2025) Group B

Task 1: Short questions 15 points

1.1 We are measuring the memory requirement of a new data structure developed in Java. We notice two things:

- We can run all the measurements with a 1 GB memory limit, but we measure high runtimes caused by the garbage collection mechanism of Java.
- On the other hand, if we run all measurements with 32 GB of memory, we measure low runtimes but high memory usage (22 GB of memory usage is reported by the profiler we use).

Explain the inconsistencies in the measurements and propose a way to evaluate the memory requirements.

1.2 We evaluated the memory requirements of our new data structure ten times each on five increasingly difficult problem sizes, and we want to visualize the results using a scatterplot diagram. Draw an example **scatterplot** with five problem sizes. Explain what happens with scatterplots if multiple measurements have the same values.

1.3 Create an example **control flow diagram** and corresponding **test inputs** where the statement coverage is 100% but the MC/DC coverage is not.

Task 2: Graph modeling 25 points

a) Write a **Refinery metamodel** based on the following specification:

We want to model a system for managing the reconstruction and translation workflow of ancient texts. Each ancient text fragment may be stored in at most 1 museum collection. A reconstruction combines 1 to 100 fragments and can be in one of three states: proposed, validated, or disputed. A translation project works on exactly one accepted reconstruction and use least 1 translation aid. Dictionaries are translation aids for exactly 2 dialects. AI models are translation aids that are trained on at least 1 dialect. They can either be recurrent neural networks or transformers. Each dialect is studied by at least 1 scholar and all translation projects must be peer-reviewed by at least 2 scholars. Moreover, each museum collection is curated by exactly 1 scholar.

Only provide Refinery code and do NOT write Java code or draw an UML class diagram.

Use the following concept names: `AcceptedReconstruction`, `AIModel`, `curatedBy`, `Dialect`, `dialects`, `Dictionary`, `DisputedReconstruction`, `fragments`, `MuseumCollection`, `ProposedReconstruction`, `Reconstruction`, `RecurrentNeuralNetwork`, `reviewedBy`, `Scholar`, `storedIn`, `studiedBy`, `TextFragment`, `trainedOn`, `Transformer`, `TranslationAid`, `TranslationProject`, `uses`, `worksOn`

b) Draw a **graph model** based on the following data:

Fragments B3, B4, and B5 of the Dresden Codex are stored in the Dresden State Art Collection. They are part of an accepted reconstruction called the "Dresden Almanac Section". The Dresden Weather Passages translation project works on this reconstruction using a transformer AI model called MayaGPT that is trained on Classical Maya and Ch'olti' dialects. They also use a Classical Maya – Colonial Spanish dictionary. The project is reviewed by two scholars: Prof. Lisa Chen (the curator of the Dresden collection) and Dr. Maria Rodriguez. Both reviewers study Classical Maya and Ch'olti'. Dr. Rodriguez also studies Colonial Spanish.

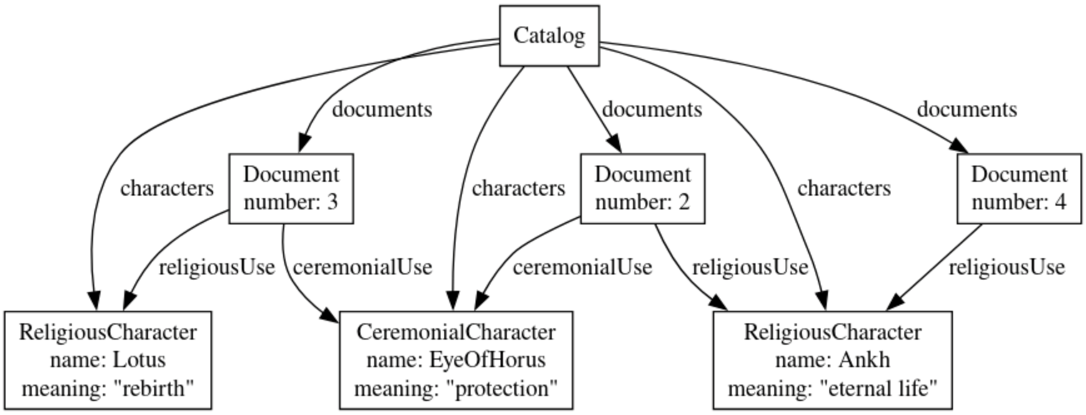
Only provide a graph model and do NOT write Java or Refinery code.

c) According to a domain expert, the following **constraint** holds: *a reviewer must study at least one dialect involved in a translation project*. However, the expert suspects that this cannot be enforced by the current version of the metamodel. To confirm, draw a **graph model** that conforms to the metamodel but violates the constraint.

d) According to the domain expert, some translation projects use proposed or disputed reconstructions. **Recommend** a way to update the metamodel for representing such projects.

Task 3: Textual modeling **30 points**

We would like to create a **textual domain-specific modeling language** to describe the meanings and usage of hieroglyphic characters. The example below shows the desired **concrete syntax** (textual description) **and abstract syntax** (instance graph model) for the language:



```
characters:
- Lotus means "rebirth" and is a religious character
- Ankh means "eternal life" and is
  a religious character
- EyeOfHorus means "protection" and
  is a ceremonial character

documents:
- Tablet 2 uses Ankh and EyeOfHorus
- Tablet 3 uses Lotus and EyeOfHorus
- Tablet 4 uses Ankh
```

a) Create a **Langium grammar** to parse this language! These declarations are available to you:

```
grammar Hieroghyphic

hidden terminal WS: /\s+/;
terminal ID: /[_a-zA-Z][\w_]*/;
terminal INT: /\d+/;
terminal STRING: /"[^"]*"/;
```

Provide the rest of the grammar.

b) Create a **Jinja2 template** an SQL script to insert the data into a relational database. The input of the template is the `Catalog` object parsed by the grammar you created in part a). An example SQL script is shown below:

```
INSERT INTO characterUses (documentId, character, meaning)
VALUES
(2, 'Ankh', 'eternal life')
(2, 'EyeOfHorus', 'ceremonial')
(3, 'Lotus', 'rebirth')
(3, 'EyeOfHorus', 'ceremonial')
(4, 'Ankh', 'eternal life')
```

To help you in reading the example, we emphasized text coming directly from instance model in **bold**. For ceremonial characters, output the `meaning` value `'ceremonial'` instead of the one in the instance model. You may assume that the name or meaning of each character is a valid SQL string that does not contain the special character `'` (single quote).

The `type` attribute contains the type of each object (e.g., use `x.type == "ReligiousCharacter"` to check if `x` is of type `ReligiousCharacter`) in the instance model. Cross-references are encoded as strings equal to the `name` of the referenced object, while missing cross-references (with no value set) are omitted from the model objects.