Formal Methods in Software Engineering
Assignment on *Alloy*

This assignment is about modeling a library management system using Alloy. Create a file "`library-`*your-name*`.als`", and include in it the following code:

```
sig Book {}
sig User {borrowed: set Book}
sig SeniorUser extends User {}
```

Problem 1. Add a predicate `BorrowRules` (with no parameters) that checks that (1) each book is borrowed by at most one user, and (2) each user who is not a `SeniorUser` has borrowed at most one book. (A `SeniorUser` can borrow any number of books.) Write a fact that enforces `BorrowRules`. Also, include the following commands:

```
pred Show {}
run Show for 9
```

Make sure your system is not over-constrained. That is, *all* solutions that satisfy the two rules mentioned above should be instances of the model.

Problem 2. In this part we will add "operations" to the model.

(1) Model a *borrow* operation as a predicate `borrow`, taking parameters `u:User`, `b:Book`, and `borrowed1:User -> set Book`. Write the predicate such that when it is run, Alloy will find solutions of the form (instance i, and values for `u`, `b`, and `borrowed1`) such that `borrowed1` represents the result of `u` attempting to borrow `b` when the system is in the state represented by i. Ensure that `u` can borrow a book `b` only if `b` is not currently borrowed by any one in the instance i.

(2) Model a *return* operation as a predicate `return`, taking parameters `u:User`, `b:Book`, and `borrowed1:User -> set Book`. This operation models the return of a book `b` by user `u`. After it is returned, the book `b` is to be not borrowed by any one (in `borrowed1`). Ensure that the operation does not change the borrowed status of `b` if `u` has not borrowed `b` in the instance i.

Some requirements from your solution are as follows:

- Include a `run` command for each of the two predicates above. (That is, you should let Alloy find solutions to the parameters required by the predicates.)
- Each solution found by Alloy for a predicate above intuitively represents an original instance (before the operation) as well as an updated instance (after the operation). Write each predicate such that the updated instances always satisfy the two rules mentioned under Problem 1.
- Don't directly check in your predicates whether the updated instance satisfies the rules. Rather, check the *pre-conditions* on the *original instance*, and on `u` and `b` that are necessary and sufficient to ensure that the updated instance satisfies the rules.

- Your predicates should be relaxed enough that Alloy can enumerate solutions representing *all* possible scenarios (wherein the updated instance satisfies the rules). For example, a person who has already borrowed a book attempting to borrow the same book, a person returning a book that they don't currently hold, etc.
- For each predicate X that you wrote above (i.e., X is `borrow` or `return`), write a wrapper predicate `verify`X with the same parameters as X, which invokes X and then checks whether the updated instance fails to satisfy any of the two borrowing rules. (These wrapper predicates should generate no solutions when run.)

Use meaningful names for the various identifiers you introduce. Write a comment for each construct you write, explaining in a sentence or two what it does, the meaning of its formal parameters, etc. Include your entire model in the file "`library-`*your-name*`.als`" and email it to `raghavan@iisc.ac.in`.

Solutions to the following problems are to be written by you on paper.

Problem 3. As an extension to the library model developed above, propose a way to associate a unique language with each book. Write the necessary signatures, relations, and facts to enforce this.

The following two problems are regarding the genealogy example developed in class.

Problem 4. Assuming a domain size of 2 for `Person`, and using the procedure discussed in class for translating Alloy expressions to propositional formulas, provide a translation for the following Alloy expression:

```
all p: Person | (p in Adam) or (some q: Person | q in p.parents)
```

Write the matrix or formula for each sub-expression of the full expression, as we have done in the lecture slides.

Problem 5. Assuming a domain size of 3 for `Person`, show a solution (i.e., assignment of true or false to each boolean variable) that corresponds to the instance shown in Slide 7 of the class slide deck (the slide titled "Solution to an Alloy model"). Also, for each entity in the instance, indicate the index of that element in `Person` set (index is 0, 1, or 2).