EECS3311 Winter 2019
Guide to LabTest 1 (for Sections M and Z)

1 Rules

– You must show up for your scheduled Lab session.

– Bring a piece of photo ID.

– No mobile phone usage is allowed during the test. The Labtest will be in Exam mode, and thus academic integrity rules apply.

– No data sheet will be allowed.

2 Coverage

– Review and understand all lectures, required readings, and Lab1, Lab2, Lab3 and Lab4. Notice that we covered the singleton and iterator `design patterns`. We are likely to require the use of one of these design patterns in the Labtest.

– By now, given the intensive scheduled Lab work, we assume that you are comfortable with Eiffel syntax, the IDE, Design by Contract, unit testing, and the use of the debugger. You will need these competencies in the Labtest.

– We used the `Mathmodels` specification library in Lab1, Lab2, and Lab3.

– Specifically review the example of Birthday Book discussed in class. For section Z, see the SVN. You may also use the section M videos and materials:

  – `https://www.eecs.yorku.ca/~jackie/teaching/lectures/index.html#EECS3311_W19`
  – Abstractions via Mathematical Models. `Mathmodels` is covered in Lectures 10 and (the first part of) Lecture 11.
  – See also the lectures on Complete Contracts
  – We have made the design and implemented code of birthday book available in `https://github.com/yuselg/eiffel/tree/master/snippets/birthday-book`

– In this Labtest, you will be designing and implementing an example similar in scope to the birthday book.

  In the birthday book the model was a **function**: `model: FUN [NAME, BIRTHDAY]`. One might write a specification: `model ~ (old model.deep_twin @<+ [a_name, d])`. The infix operator is an alias for the immutable query `overriden_by (t: TUPLE[g: G; h: H]): FUN[G,H]`.

  In the Labtest, you will be using a **relation** (`REL [G, H]`) in the specification of the design. Relations have many of the operators of functions, including the override operator. But naturally relations also work differently. You should investigate relational operators such as:

  – +: extended_by

  - `@<<`: domain_subtracted_by
  - `@<`: domain_restricted_by
  - Relational image using the alias `[]`. In a relation, an element in the domain may have many related elements in the range, Thus the image (`image alias "[]" (g:  G): SET[H]`) returns a set. For example, given a relation $r = \{a \mapsto 1, b \mapsto 2, a \mapsto 3\}$, then $r[a] = \{1, 3\}$.

    Notice that in Mathmodels, a relation is defined as: `SET[PAIR[G, H]]`. Such an unrestricted definition will not work for `FUN[G,H]`. Why? Our different design of classes `FUN` and `REL` reflect the underlying mathematical reality.

3 Format

- There will be **no** written parts for this test.
- Your marks will be determined entirely by the number of tests that your compiled code/design passes. No partial marks will be given to code that does not compile. Compile and test often.
- You will be coding contracts and implementations in EStudio.
  - We assume familiarity with: arrays, lists, the `across` notation, and classes in `mathmodels` library (particularly the immutable queries of classes `SET`, `REL`, and `FUN`).
  - It is important to ensure that (1) all contracts are specified in terms of the immutable queries of Mathmodels, and (2) all implementations may not use Mathmodels, but must be encoded in terms of efficient data structures such as arrays, lists, etc.