CSE3311 SOFTWARE DESIGN – ASSIGNMENT 1
MAP ADT AND IMPLEMENTATION
VER. 1.1

PRZEMYSLAW PAWLUK

Due: Wednesday, May 29, 2013

Where: In class

Weight: 10%

1. Main Points

Be sure to read and follow all the guidelines from the links on reports and academic honesty from the WWW home page for the course. The specification is the union of this document plus the program text you are given.

1.1. Learning objectives.

- The implementation and documentation of abstract data types as classes
- Eiffel programming
- Programming from contracts and test cases

1.2. **To hand in.** Hand in, in class, a report containing the following items as a package in the given order. If the package is too big to staple as one unit, then please staple in multiple parts. Cover page as part *1 of N*, the remaining as part *x of N*. Include your name(s) on each part. The report is required for your work to be evaluated. The electronic submission is used to run your system but with no report, the electronic submission is ignored.

(1) Cover page printed from the course web pages
(2) A report describing the ADT. The structure is to be based on the slide set Abstract Data Types Documentation.
(3) A listing of the file `my_map.e` should be the last section in your report. A 10 point fixed width font such as courier should be used the program enscript on Prism is an ideal choice.

1.3. **Electronic submission.** Before the deadline, submit all .e and .ecf files for the system. Use relative addressing in your system so your program will compile no matter where the files exist on Prism or Windows systems. No other files should be submitted (e.g. EIFGEN files etc). You should use **eclean** (on Prism) before submitting to clear away the unnecessary files. To submit, use the following command on Prism.

```
submit 3311 a1 mymap.ecf *.e.
```

Files cannot be deleted the submit command can only add or replace files so be very careful to clean up your directory before any submission. While you can develop your system on your personal computer, be sure your program will compile and execute on Prism using estudio72

1.4. **To get started.** Download the file 3311_a2.zip to your local directory. Decompress it. You will get a directory called `my_map` that contains all files required for the project.

2. Tasks

You are to complete the missing parts in the file `map.e`. The given partial implementation is a generic structure of Map, where different instances can store objects of different types but a particular instance will store only sequences of the same type of objects. Objects are inserted in the map using the put feature. For simplicity, the argument passed to the put feature is a pair of key and value – object in the given sequence.

Missing parts are marked as `--??`. Surround each added part of the code with a comments `--begin added` and `--end added`.

3. Definition of a Map

In computer science, a map (called also an associative array or dictionary) is an abstract data type composed of a collection of pairs $(key, value)$, such that each possible key appears at most once in the collection. Operations associated with this data type allow:

- **put** – the addition of pairs to the collection
- **remove** – the removal of pairs from the collection
- **get** – the lookup of the value associated with a particular key

Requirements:
If **put** is called for a key that exists in the map, value associated with this key in the map is overridden. If **get** is called with a key that does not exist void is returned. Performing **remove** with a key that does not exist does not change the state of the map.

You will have to implement all helper functions required to pass the tests such as quantifiers.

You have to provide contracts for each feature (require and ensure clauses) and the class (invariant). You should provide as many contracts as you can implement in Eiffel. Those, that you consider necessary but don't know how to implement should be written in a natural language (english) as comments in the place where they belong.

4. Grading scheme

The grade for the report is partitioned into the following parts.

- Overall presentation 10%
- `MY_MAP` ADT description 25%
- Programming (all tests must pass) 40%

  - Contracts – 20%
  - Implementation – 20%
- Comments in `my_map.e` 25%