# CSE 3311 SOFTWARE DESIGN
# REPORT 1 SPECIFICATION

PRZEMYSLAW PAWLUK

**Due:** Wednesday, September 27, 5:30pm
**Where:** In class
**NOTE:** If the class has begun your report is late

## 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 dict.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 r1 dict.ecf *.e.
```

*Date*: September 12, 2012.

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 estudio 7.0.

1.4. **To get started.** Download the file `3311_report1.tar.gz` to a local directory. When you untar with the command `tar xzf 3311_report1.tar.gz`, you will get a directory called report1 that contains the following files.

(1) `test_dict_all.e` the class TEST_DICT_ALL uses eSpec to test the dict system. Do not modify this file.
(2) `test_dict_char.e` the class TEST_DICT_CHAR uses eSpec to test character tries. Do not modify this file.
(3) `test_dict_string.e` the class TEST_DICT_STRING uses eSpec to test string tries. Do not modify this file.
(4) `dict.e` the class DICT is an implementation of a dict ADT. You are to complete the implementation of the routine bodies, where indicated.
(5) `dict.ecf` the Eiffel construction file for the dict system. Do not modify this file.

The system will compile and run but most of the tests will fail. The tests that pass, pass for the wrong reason (an example that tests may not necessarily test what you think they test).

## 2. Tasks

You are to complete the missing parts in the file dict.e. The given partial implementation is a generic dict, where different instances can store sequences of objects of different types but a particular instance will store only sequences of the same type of objects. Sequences are inserted in the dict using the insert feature. For simplicity, the argument passed to the insert feature is a linked list of the objects in the given sequence (in a real application this may be a more generic container).

## 3. Definition of a Dict

A dict is a data structure that can be used to store sequences of elements in a manner that makes it efficient to check whether a particular sequence is contained in the dict. Its most common application is in storing words (sequences of characters). Assuming all valid English words are stored in a dict, a spell-checking program can quickly check if a word is valid or not. For example, Figure 1 shows a dict containing the words he, hers, his, hop, hope and she, while the words her and hi are not in the dict. A spell-checker examining the word scan quickly see that there is no sequence starting with the character e and conclude that it is an invalid word.

Each node, see Figure 2, in the dict contains an element, a character in the example in Figure 1, and a flag, that indicates whether the sequence starting at the root and ending at this node is valid or not. The root node of the dict does not contain an element.

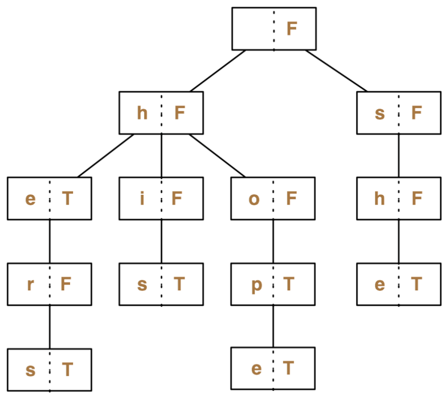

Figure 1. An example of dict

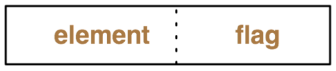

Figure 2. An abstract dict node

## 4. Grading scheme

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

- Overall presentation 10%
- Trie ADT description 25%
- Programming 40%
- Comments in dict.e 25%