CMSI 585

Programming Languages (Graduate Level)

Fall 2006

Assignment 1031

This assignment continues with control flow exercises, now focusing on loops and recursion.

Not for Submission

At this point, we have now covered all of the primary points of interest in Scott Chapter 6.

For Submission

Commit the programming tasks under the specified directories with the given tag, and submit your answers to the assigned textbook exercises and other questions on hardcopy. If you do the extra credit work, submit that on hardcopy as well. The extra credit is cumulative, not selective — the "extra extra" credit only counts if you've already done the "single extra" credit exercise.

1. Answer Scott Exercise 6.12.
2. Consider a potential programming language feature that allows strings in a *case*/*switch* statement, as shown below in C-like syntax:

   ```
   String s;
   ...
   switch (s) {
     case "Hello": ...
     case "Goodbye": ...
     default: ...
   }
   ```

   What are the potential advantages and disadvantages of such a feature? Include both language design and implementation perspectives.
3. Implement the *doLoops()* method that is stubbed out in the *loop.Loop* handout. The Javadoc comment for that method describes what it is supposed to do.
4. Write a JUnit unit test that shows the result of this *doLoops()* method by calling your implementation of *doLoops()* then stating the expected contents of the returned *List<Loop>*. The test class should have the fully-qualified name *loop.test.yourLastName_LoopTest*.

   Commit the Java code under */homework/cmsi585/loops/java*. Tag the files with *hw-1031*.
5. Program Scott Exercise 6.27 in ML, with the following specifications:
   - Place the *factorial* function itself in a file called *yourLastName_factorial.sml* (note: that is just the filename; the function itself should retain the name *factorial*).
   - Create a unit test for your *factorial* function in a file called *yourLastName_testFactorial.sml*. Use the "homemade" ML unit test functions provided in the *assertFactorial.sml* handout to make your assertions.

   Commit this code under */homework/cmsi585/tailrecursion/ml*. Tag these files with *hw-1031*.
6. How would one determine whether or not a programming language implementation detects and transforms tail recursions?
7. Answer Scott Exercise 6.32.

Extra Credit

The "for-loop-over-a-collection" constructs of Java and JavaScript are syntactically very similar, but semantically *very* different:

```
/* Java */
String[] stringList = {"Hello", "Goodbye"};
for (String s: stringList) { ... }

/* JavaScript */
var stringList = [ "Hello", "Goodbye" ];
for (var s in stringList) { ... }
```

Look up these constructs in each language, writing little test programs if necessary, and state how they differ from each other semantically.

Extra Extra Credit

Do some research on the features of any programming language that is not in our "big 6" (C, C++, Java, JavaScript, ML, and Perl) and describe the range of loop constructs in that language.