CMSI 585

PROGRAMMING LANGUAGES
(GRADUATE LEVEL)
Fall 2005

Assignment 1018

This assignment seeks to give you a firsthand (that is, programming) feel for some key concepts with regard to naming in programming languages — about which there will be at least one midterm question. So, despite the deadline, at least try your hand at it as part of your midterm preparation.

Not for Submission

If you haven't done so already, install Perl and its associated unit test modules (Test::Simple, Test::Class, Test::More, Test::Exception) on your working system.

For Submission

As usual, submit your code on hardcopy and by e-mail. Write a Perl module called NameGame that mimics the sample JavaScript NameGame code. The NameGame module should contain the following:

1. A statically-allocated scalar variable, with initial value 99. You can name it anything you want, but for brevity, we will call it *total* here. Then define the subroutines below.
2. *getOutsideTotal* — returns the statically-allocated *total*
3. *setOutsideTotal* — sets the statically-allocated *total* to the first argument
4. *addToOutsideTotal* — adds the argument to *total* and returns *total*
5. *addToLocalTotal* — defines a statically-scoped local variable called *total*, sets the initial value to 20, adds the argument to that local variable, then returns it
6. *addToDynamicTotal* — defines a *dynamically-scoped* local variable called *total*, sets the initial value to 50, then relays the subroutine's first argument to *addToOutsideTotal*
7. *addToNestedTotal* — takes 2 arguments: the first is a value to add (in this example, let's call it *addend*), and the second is expected to be a subroutine reference that takes a single scalar argument. Define a *nested subroutine* (let's call it *nestedAdd* here), which returns the value of its argument minus *addend*; then, in the main body of the subroutine, if *addend* > 50, we relay *addend* to the referenced subroutine, else we re-invoke *addToNestedTotal* with arguments 51 and a reference to *nestedAdd*. (it's not as complicated as it sounds — most of the logic is already in the JavaScript version)

Next, write a Perl test module called *lastName_NameGameTest* that performs the following:

8. A standard *setup* subroutine that sets the outside total in NameGame to 99.
9. A set of test subroutines that tests the functionality of the 6 subroutines in NameGame. You can pattern these tests after the JavaScript test suite, but don't use the same numbers (after all, if we all used the same numbers, then what good would different tests be?).

We will use the same unit test/open source technique to see how we do in writing this code. As always, don't hesitate to e-mail me with any questions or clarifications.