CMPS 12L
Introduction to Programming Lab
Winter 2008

Lab Assignment 4

Due Wednesday February 20, 10:00 pm

Our goal in this lab assignment is to test a slightly buggy (but different) version of the `LetterHome` program from Lab Assignment 3. This time however, we will not be debugging the program, but simply testing it. The distinction is that in testing, we seek to reveal the existence of bugs, whereas debugging is meant to eliminate them once they have been identified. Debugging is a process of correction, while testing is a process of evaluation, and may be subjective to some degree. To debug, one must have access to the program source code, but this need not be the case during testing. A software tester may have only the program's specification (i.e. a complete description of the program's intended operation) to go by.

Software testing may be defined as the process used to verify the correctness, completeness, security, and quality of programs. This process generally includes, but is not limited to, a set of program executions on strategically chosen inputs, with the intent of finding errors. How these inputs are chosen often depends on who is doing the testing and why. Typically, when a programmer sets out to test his or her own work, a collection of inputs is chosen that will exercise every statement in the program, by following every possible logical pathway the program can take. This method of testing is called *branch coverage*, and it belongs to a set of testing regimes known as *white box testing*, since one must see the source code to design such inputs. You should carry out this form of testing on all of the programs you write for this class.

Often though, software is tested by the user, or by someone working on the user's behalf. In this setting, typically, the source code is proprietary information belonging to the developer, and is therefore not available for inspection. The testing regimes used in this scenario are known collectively as *black box testing*. In this case, the tester must choose a set of program inputs designed to verify all program specifications. Ideally, one checks that the program behaves as intended, in all conceivable circumstances. Unfortunately this is usually impossible, since the entire set of possible program inputs is very large, if not infinite. This is where subjectivity may come in. One must focus on those inputs which are most closely related to the program's critical functions, and which are most likely to arise in practice.

In this assignment, you will perform black box testing on the `LetterHome` program. The specification for this program is the *general template for the letter body*, given in the comment block of the file `LetterHome.java` in the examples section of the website. Furthermore, it is specified that all program inputs are integers, and that a properly formatted input file consists of any number of lines, each containing two integers separated by a space. If the sentence code (the first integer in a pair) is out of range, the program is required to print an error message, while if the modifier code (the second integer) is out of range, the program must print the word "ERROR" in place of the corresponding modifier. Note that it is not considered an error for the student to request a negative number of dollars in sentence 4. The program is not required to behave in any particular way on files that are wrongly formatted, so your suite of test inputs need not include files containing doubles, non-numeric strings, or any other type of wrong input.

Since this is black box testing, you should pretend that you have not actually seen the source code file `LetterHome.java`, given in lab3. In fact, the source has been changed in some critical ways, of which you are unaware, so you don't really have to pretend. You will not be given the altered source, but

instead, you will have only the corresponding executable object file `LetterHome.class` to work with. You should copy this file from `/afs/cats.ucsc.edu/courses/cmps012a-pt/` to your own account space. Begin by creating a directory within your `cs12a` directory called `lab4`, then `cd` to that directory and do

```
% cp /afs/cats.ucsc.edu/courses/cmps012a-pt/LetterHome.class  .
```

Be sure to include that final dot, which means "the directory I am now in". This command copies the object file `LetterHome.class` to your `lab4` directory, giving it the same name. Be sure not to copy this file to your `lab3` directory, or wherever you keep your corrected object file from that assignment, since the `cp` command will overwrite any file having the same name. The point is that `LetterHome` in this assignment is a different program than `LetterHome` in `lab3`.

Once you have the executable, design an input file (or files) to thoroughly test each aspect of the specifications outlined above. Notice that, even for this simple program, the set of possible inputs is infinite, since the number of dollars requested by the student in sentence 4 could be any integer (even a negative one) . This poses a problem for you, the tester, since for all you know, the program will crash when the dollar amount 6732 is entered, and for no other amount. You must use your best judgment in selecting these test inputs. Your test file(s) should elicit a number of errors from the program, which are defined to be any departure from the above specifications. Look closely at the output since these could be just minor grammatical errors. Create a text file called `errors` containing a numbered list of the errors you found in the executable `LeterHome.class`. Include a short one or two sentence description of each error. Your description should in no way refer to the source file for the program, since after all, you don't (really) have that. Instead, just say in what way the specifications were violated.

Submit your file `errors`, along with any test files you created for this assignment to the `lab4` submit directory:

```
%  submit  cmps012l-pt.w08 lab4  errors  test_file1  test_file2  . . .
```

You may find this to be one of the easiest labs to complete, but it will take some time to do it carefully, so please do not wait until the last minute to begin.