Homework 1: Introduction to Programming Concepts

General Directions

Please see the course website (`http://www.eecs.ucf.edu/~fhussain/cop4020/`) for due dates.

Answers to English questions should be in your own words; don't just quote text from the textbook.

For programming problems for which we provide tests, you can find them all in `hw1tests.zip`. If the tests don't pass, please try to say why they don't pass, as this makes commenting on the code easier and more specific to your problem.

How testing works

This section briefly explains how our tests work. You do not need to understand the details explained here in order to complete the homework. However, you will need to run our tests and show the Oz emulator output (which can be done in emacs using C-. e).

Our tests use the file `TestingNoStop.oz`. The `Test` procedure in this file can be passed an actual value, a connective (which is used only in printing), and an expected value, as in the following statement.

```
{Test {CombA 4 3} '==' 24 div (6*1)}
```

The `Assert` procedure in this file can be passed a boolean, as in the following statement

```
{Assert {Comb J I} == {CombB J I}}
```

Calls to `Assert` produce no output unless they are passed the argument **`false`**. Note that you would not pass to `Browse` or `Show` a call to `Test` or `Assert`, since neither of these procedures returns a value. **If you're not sure how to use our testing code, ask us for help.**

What to turn in

For problems that require code, you must turn in both: (1) the code file (use text files with the name given in the problem or testing file or its main function and with the suffix `.oz`) and (2) the output of running our tests using your file.

Please turn in test output and English answers as plain text files with suffix `.txt`. Please do not put spaces or tabs in your file names.

Your code should compile with Oz; if it doesn't you should keep working on it. **Email the staff with your code file if you need help getting it to compile.**

How to submit

Please submit a `zip` or `tar.gz` format file that contains all your files (do not include our testing files). Please name the file using your full name as: `FirstnameLastnameHW1.zip` (or `FirstnameLastnameHW1.tar.gz`) and email it to `newton@knights.ucf.edu`.

Extra credit problems are to be turned in separately. Please submit a `zip` or `tar.gz` format file that contains answers to all the extra credit problems you're attempting. Please name the file using your full name as: `FirstnameLastnameHW1EC.zip` (or `FirstnameLastnameHW1EC.tar.gz`) and email it to `fhussain@eecs.ucf.edu`.

Other directions

You should use helping functions whenever you find that useful. Unless we specifically say how you are to solve a problem, feel free to use any functions from the Oz library (base environment).

Please do not hesitate to contact the staff if you are stuck at some point.

For background, read Chapter 1 of the textbook [VH04]. You may also want to refer to the reference and tutorial material on the Mozart/Oz web site.

Reading Problems

The problems in this section are intended to get you to read the book, ideally in advance of class meetings.

Read §1.1 and §1.2 of the textbook [VH04] and answer the following questions.

1.
   (a) (2 points) What does `{Browse funny}` do in Oz?
   (b) (3 points) What kind of character must a variable identifier, start with in Oz?
   (c) (5 points) Can variables in Oz be assigned a value more than once? (Answer "yes" or "no" and give a brief explanation.

Read §1.3-§1.15 of the textbook and answer the following questions.

2. (5 points) This question deals with lists that represent "3 address instructions." Such lists always have 4 elements and look something like `[add 2 5 7]` where the symbol `add` tells what instruction it is, and the numbers 2, 5, and 7 stand for register numbers or addresses. These lists will always have exactly 4 elements. Using Oz's pattern matching feature, write a function that returns true just when the second and third elements of such a list are equal (i.e., compare equal using ==), and false otherwise. Call this function `First2OpsEqual`. The following are some tests, found in the file First2OpsEqualTest.oz; you should run these tests on your code and hand in the output along with your code (as described above).

   To run our tests, put your file `First2OpsEqual.oz` and our test file `First2OpsEqualTest.oz` in the same directory. Then run our tests by feeding the buffer `First2OpsEqualTest.oz` to Oz. You will have to look at the `*Oz Emulator*` buffer to see the output.

```
% $Id: First2OpsEqualTest.oz,v 1.1 2010/08/12 00:40:17 leavens Exp leavens $
\insert 'First2OpsEqual.oz'
\insert 'TestingNoStop.oz'
{StartTesting 'First2OpsEqualTest $Revision: 1.1 $'}
% If you fix your code, then you may have to restart Oz to make these pass...
{Test {First2OpsEqual add|2|7|5|nil} '==' false}
{Test {First2OpsEqual add|3|3|5|nil} '==' true}
{Test {First2OpsEqual mult|3|3|8|nil} '==' true}
{Test {First2OpsEqual divide|0|1|8|nil} '==' false}
{Test {First2OpsEqual sub|1|1|8|nil} '==' true}
{StartTesting 'done'}
```

   Hint: Note that you can program this with a single pattern match in a single **case** expression. Note: Be sure to put your code in a file named `First2OpsEqual.oz`, otherwise our testing code won't find your solution. Our tests only call `First2OpsEqual` on lists that are 4 elements long, so you should assume that the inputs will have that type.

   You are prohibited from using the numeric field dereference operators, such as `.1` and `.2`, in your solution.

3. (5 points) Briefly explain what happens when the following code executes in Oz?

```
local SetIt It IgnoreVal in
   It = good

   fun {SetIt X}
      It = X
      true
   end
   IgnoreVal = {SetIt bad}
   {Browse 'It is: '#It}
end
```

4. (5 points) Briefly explain what happens when the following code executes in Oz?

```
local X in
   X = X+1
   {Browse 'X is '#X}
end
```

Read §1.3-§1.15 of the textbook and answer the following questions.

5. (5 points) According to chapter 1, what problems does nondeterminism cause in concurrent programming?

Regular Problems

We expect you'll do the problems in this section after reading the entire chapter. However, you can probably do some of them after reading only part of the chapter.

The textbook problems are from the *Concepts, Techniques and Models of Computer Programming* book [VH04, §1.18].

6. (20 points) Do problem 2 in chapter 1, calculating combinations. Note that this should be done without using cells or assignment (that is, in the declarative model).

   Your solution's Oz code should be in a file `Comb.oz`, and that file should contain two functions. Part (a)'s solution should be called `CombA`, and part (b)'s solution called `CombB`.

   Hint: use the function `Comb` from §1.3, and use recursion. Don't write the same code twice, instead make function calls. In Oz, use **`div`** to divide two integers; for example, `12 div 4` returns 3.

   You must test your code using Mozart/Oz. After doing your own tests (with `Show` or `Browse`) you must run our tests. To do this, put your file `Comb.oz` and our test file `CombTest.oz` in the same directory. Then run our tests by feeding the buffer `CombTest.oz` to Oz. You will have to look at the `*Oz Emulator*` buffer to see the output.

   If you have trouble running our tests, see the troubleshooting section of the course's running Oz page. If that doesn't help, contact the course staff.

7. (10 points) Do problem 5 in chapter 1, lazy evaluation.

8. (10 points) Do Problem 7 in chapter 1, explicit state.

9. (15 points) Do problem 10 in chapter 1, explicit state and concurrency.

Extra Credit Problems

Please note that extra credit problems are entirely optional. Please do not spend excessive time on these at the expense of other (compulsory) questions.

10. (20 points; extra credit) Do problem 3 in chapter 1, correctness of `Pascal` function.

11. (15 points; extra credit) Do problem 8 in chapter 1, explicit state and functions.

Points

This homework's total points: 85. This homework's total extra credit points: 35.

References

[VH04] Peter Van Roy and Seif Haridi. *Concepts, Techniques, and Models of Computer Programming*. The MIT Press, Cambridge, Mass., 2004.