

CIS-280
Homework 8:
Due Thursday, April 28, 2005
75 points

1. (25 points) **Message-passing:** We can represent numbers as integers (35), decimals (56.34) or as floating point numbers. We will only be concerned with numbers that are larger than 1. We want to be able to get a number's mantissa (message = Mantissa), exponent (message = Exponent), least-upper-bound (message = LUB), and whether it is greater than or equal to 1000 (message = GEQ-1000). If the number is of type integer or decimal, then the mantissa and exponent is what they would be if the number were changed to mantissa/exponent form. We will not care about the number of digits in the mantissa, only with having a mantissa that is $\geq .1$ and < 1.0 . Use the message passing style of programming to design 1) procedure (`make-float mt e`), where `mt` is the mantissa (which is $\geq .1$ and < 1.0) and `e` is the exponent, 2) procedure (`make-decimal d`) where `d` is a decimal number, and 3) procedure (`make-integer int`) where `int` is an integer.
2. (15 points) **Message Passing:** Write a procedure (`Compute message . number`) that will take a message (such as Mantissa) and return a list of the mantissas of each number in the argument list, where the first element of the resultant list corresponds to the requested entity for the first number, the second for the requested entity for the second number, etc. Note that procedure `Compute` can take an arbitrary number of arguments. **Procedure Compute may not use any helper procedures** — **Hint: use map.**
3. (35 points) **Tower of Types:** Assume that we have three types: Complex, Decimal, and Integer. You have complex numbers with constructor and selectors (`Create-complex r i`), (`Get-real-part c`) and (`Get-imag-part c`), decimal numbers with constructor and selector (`Create-decimal d`) and (`Get-decimal n`), and integer numbers with constructor and selector (`Create-integer n`) and (`Get-Integer n`). You also have procedure (`Adjoin-Type type rep`) for taking a type and a representation of a number and producing an object with manifest type, procedure (`Get-Type n`) for getting the type of a number with manifest type, and procedure (`Get-Rep n`) for taking an object with manifest type and producing the representation of the object with the type stripped off. The constructors do not assign manifest type to the number they create, but you will be working with numbers that have manifest type and your results should have manifest type. The following is a version of problem 2.85 from the textbook. You will need to load the procedures in `~carberry/HW-8.scm`
 - (a) (5 points) Write procedures (`Drop-complex c`) and (`Drop-decimal r`), that lower an object with manifest type one level in the tower. Your procedures should lower the object even if it changes the objects value. For example, procedure `Drop-complex` should lower the complex number $3.7+4i$ to the real number 3.7. In the case of `Drop-decimal`, you should truncate the decimal number to get an integer, rather than rounding it. The argument will have manifest type and the new number should have manifest type.
 - (b) (5 points) Design procedures (`Equ-complex? n1 n2`), (`Equ-decimal? n1 n2`), and (`Equ-integer? n1 n2`) that take two arguments of the same type (both with manifest type) and returns true if they are equal.
 - (c) (5 points) Write procedures (`Raise-decimal r`) and (`Raise-integer i`) that raise their argument one level in the tower. The argument will have manifest type and the new number should have manifest type.
 - (d) Write a procedure `Install-280` that uses the `put` operation to install your procedures in a table. `Drop-complex` and `Drop-decimal` should be entered into the table with column as the type and row as the operation `Drop`. `Raise-decimal` and `Raise-integer` should be entered into the

table with column as the type and row as the operation `Raise`. `Equ-complex?`, `Equ-decimal?`, and `Equ-integer?` should be entered into the table with column as the type and row as the operation `Equ?`.

- (e) (20 points) Write a procedure (`Drop number`) that takes a number with manifest type and lowers it as far as possible in the tower, **without changing the value of the number**. The trick here is to notice that if you lower a number and then raise it back up again, you should get the original number back again if the lowering was valid — ie., did not change the value of the number. **Your procedure `Drop` must work for any number of levels of dropping — that is, you cannot assume that the maximum number of levels of dropping is 2.**

Submit your code (**without the code from `HW-8.scm`**) on the electronic submission system for Homework-8. I will have the code from `HW-8.scm` already in the test system. Your submitted code should only define the procedure `Install-280` but not execute it.