CPSC 201: Introduction to Computer Science
Carsten Schürmann
Date: February 14, 2003

Homework 4

Due: Wednesday, February 19, 2003.

Guidelines

While we acknowledge that beauty is in the eye of the beholder, you should nonetheless strive for elegance in your code. Not every program which runs deserves full credit. Make sure to state invariants in comments which are sometimes implicit in the informal presentation of an exercise. If auxiliary functions are required, describe concisely what they implement. Do not reinvent wheels, and try to make your functions small and easy to understand. Use tasteful layout and avoid long winded and contorted code. None of the problems requires more than a few lines of SML code.

Exercise 1 In this exercise, we study the representational issues of real numbers a little further. Recall from class, that a real number is represented as a $1 + m + 1 + e$ bits, where m is the number of bits reserved for the mantissa, and e is the number of bits reserved for the exponent. The two extra bits serve as sign for mantissa and exponent, respectively.

1. Implement a structure `RealNumber` that matches the following signature:

```
signature REALNUMBER =
  sig
    type Format = int * int
    type Real = bool list
    val realToReal   : Format -> real -> Real
    val RealToreal   : Format -> Real -> real
  end
```

   A `Format` is a pair of two integers `(`m`, `e`)` that characterize how to interpret a `Real` number represented as list of $1 + m + 1 + e$ bits. `realToReal` is a function that takes a `Format` f and a real number of type `real` as argument, and computes a `Real` number described by f. `RealToreal` is the reverse function that takes a format f and a `Real` number and converts it into a number of the standard SML type `real`. *Hint: Consider first how to convert integers and reals into bits, and then the reverse.*

2. Next we consider how good our encoding of floating point numbers is. To this end we consider the smallest difference between any two of our `Real` numbers, and the maximum range. This can be established by implementing two functions, one

is called `createList` that generates *all* $2^{1+m+1+e}$ `Real` numbers from bit strings of length $1 + m + 1 + e$. This list is then passed to the other function called `minmax`, which then computes the smallest and the largest difference between any two numbers contained in that list.

```
signature TEST =
  sig
    val createList  : RealNumber.Format -> real list
    val minmax      : RealNumber.Format -> real * real
  end
```

Remark: To do this exercise, copy the file `blue4.sml` *from the course directory* `/c/cs201/lib/code/ass4` *into your personal directory. Note, that the file will not load into SML unless you have defined the structure* `RealNumber` *before declaring signature* `TEST`.