Homework 3 — ML core language — assigned Thursday 5 February — due Sunday 15 February

In all exercises, try to use higher-order functions from the `List` structure in preference to pointwise recursive function definitions.

3.1 Using lists for sets: writing recursive functions over lists (25pts)

Let us use the ML type `int list` to represent sets of integers.

- (5pts) Write an ML function `union: int list * int list -> int list` that takes two sets and returns their union.
- (5pts) Write an ML function `intersection: int list * int list -> int list` that takes two sets and returns their intersection.
- (15pts) Write an ML function `powerset: int list -> int list list` that takes a set S and returns its powerset 2^S.

3.2 Using lists for arithmetic: writing recursive functions over lists (45pts)

This is a continuation of exercise 2.4. Numerals can be represented as lists of integers. For instance, decimal numerals can be expressed as lists of integers from 0 to 9. In this representation, the integer 12345678901234567890 would be represented as the ML list `[1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0]: int list`.

Write the following functions:

- (15pts) `mulLongInts: int -> (int list * int list) -> int list`, such that `mulLongInts` r (a,b) computes the product of the nonnegative integers given by the lists a and b; all lists use the same radix r. Lists a and b can represent arbitrarily large positive integers.
- (15pts) `divLongInts: int -> (int list * int list) -> (int list * int list)`, such that `divLongInts` r (a,b) computes the quotient and the remainder from the division of the nonnegative integers given by the lists a and b; all lists use the same radix r. Lists a and b can represent arbitrarily large positive integers. You can assume that b does not represent zero.
- (15pts) `changeRadix: int -> int -> int list -> int list`, such that `changeRadix` r_1 r_2 a computes the representation in radix r_2 of the same number that is given by the list a in radix r_1.

3.3 Drawing (30pts)

Write an ML function

```
plotFunctions:
  {
   functions: (real -> real) list,
   interval: {lower: real, upper: real},
   numPoints: int
  }
  -> string
```

The argument to the function contains a list of ML `real -> real` functions that compute mathematical real functions; a real interval given by its lower and upper boundary; and the number of interior points to be used for function evaluation, spaced uniformly *between* the lower and the upper boundary. The result of the function is a string consisting of valid PostScript code.

The ML function `plotFunctions` should divide the given interval into equal-length sections and thus determine a set of points, including both the interior points and the boundaries of the interval. Then it should evaluate each of the functions given at each of these points. Finally, it should emit PostScript commands that draw a line graph of each of the functions given. Each line graph should be in a different color, but none should be black.

The function `plotFunctions` should determine the region of the plane spanned by the given functions over the given interval. If the *x*-axis passes through this region, it should be drawn in black. If the *y*-axis passes through this region, it should be drawn in black.

Optional: label the line graphs, and draw labelled tick marks on the axes.

Hint: we are attempting to write a crude version of the gnuplot utility; you may follow gnuplot's choice of line graph colors.

How to turn in

Make sure that you have thoroughly tested your code, and include all your test runs!

Turn in your code by running

~clint/handin your-file

on a regular UNM CS machine. You should use whatever filename is appropriate in place of your-file.

Include the following statement with your submission, signed and dated:
I pledge my honor that in the preparation of this assignment I have complied with the University of New Mexico Board of Regents' Policy Manual, including Section 4.8, Academic Dishonesty.