COMP 302: Programming Languages and Paradigms

Winter 2015: Assignment 2

This assignment is due on Thursday, February 12, 2015 at midnight. It must be submitted using MyCourses**.** The cutoff time is automated. Assignments submitted within the next hour will be accepted but marked late. The system will not accept any attempts to submit after that time.

Problem 1: (20 marks)
Write a function `remDuplicates: ''a list -> ''a list` which when given a list, returns a list where each element occurs only once. That is, all duplicates have been removed. (The notation ``a is used to indicate an equality type rather than any type.)

Problem 2: (20 marks)
Newton's method can be used to find the root of a function f where f is a continuous real values function of the real variable x.

Suppose that x_i is an approximation to the root. Newton's method uses the derivative of the function to produce a new approximation

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

If we have started with an approximation, x_0, this process will generally converge to an accurate approximation of the root. The process will terminate when we obtain an approximation such that $|f(x_i)| < \epsilon$, for some tolerance ϵ.

Implement a function `newton: (real -> real) *real* real -> real` which when given a function f, an initial guess x_0, and a tolerance ϵ will compute an approximate root of the function using this method.

You should define a helper function derivative, which approximates the derivative of the function.

This method is not guaranteed to converge. If your function does not find a root within 1000 iterations, it should raise the exception Diverge.

Problem 3: (20 marks)
Define a function that given `f : real -> real` and `g : real -> real` and `dy : real` (a small real value) approximates the following function:
`h : real -> real`:

$$h(x) = \int_0^x f(y)g(x+y)dy$$

You can use functions we have discussed in class or define new general functions on lists if you like but **your solution to the given problem should only be one line long**. Include definitions of the auxiliary functions you use.

Problem 4:
Consider the following datatype for mathematical expressions:

```
datatype Mathexp =
       Num of int
     | Var of string
     | Neg of Mathexp
     | Add of Mathexp * Mathexp
     | Mul of Mathexp * Mathexp
```

Your functions must use structural recursion based on the structure of this datatype definition. Other methods will not be accepted.

Problem 4.1 (20 marks)
Define a function **diff** that, given a Mathexp and a variable x will compute the derivative of the expression with respect to x. The derivative is another Mathexp. The type of diff is `(Mathexp, string) -> Mathexp`

Problem 4.2 (20 marks)
Write a function, **simplify** that performs some simple optimizations on a Mathexp. In particular, it removes any subexpressions of the form `Add(e, Num(0))` and `Mul(e, Num(1))` from the given expression, replacing them by e. It also replaces subexpressions of the form `Mul(e,Num(0))` by `Num(0)`. Of course commuting variants must also be taken care of. The simplification must be "deep", not just at the first level. That is, the expression corresponding to `(x+0)+0` must simplify to `0`, not just `(x+0)`. The type of simplify is `Mathexp -> Mathexp`