Homework 6: Standard ML and Lambda Calculus

For all SML problems, you must run your code with the SML interpreter. You must also provide evidence that your program is correct (verbal arguments or test cases or both). You should program in a purely functional style (for example, don't use `ref`).

For the SML problems involving streams (sequences), you may use either the code in

`/home/cs541/public/src/streams-lazy.sml`

which is the code presented in class and essentially Paulson's code from chapter 5, or Paulson's improved version found in `/usr/unsup/sml/lcp-book/Imperative.ML`.

Please do the lambda calculus problems by hand, and only use the lambda calculus interpreter to check your work. (Don't let the computer do your thinking for you.)

Due dates will be discussed in class.

1. (5 points) Do exercise 5.28 on page 172 of Paulson's book (add adjacent elements of sequences).
2. (10 points) Do exercise 5.30 on page 174 of Paulson's book (exponential).
3. (extra credit only) Do some of exercises 5.32 and 5.34 on page 176 of Paulson's book.
4. (12 points) Do part (a) of exercise 13 (pages 57-58) of Schmidt's book, parts (i) through (vi). Note that you are only to use the β rule for this part. There's a right parenthesis missing at the far right of part (i). Hint: it's convenient to use an editor to do this kind of work, as you can write out an expression, copy it, and apply the beta or eta rules without too much hand copying.
5. (10 points) Do part (b) of exercise 13 (page 58) of Schmidt's book.
6. (extra credit) Do part (d) (i) of exercise 13 on page 58 of Schmidt's book.
7. (5 points) Using the equational theory of the simply typed calculus, show formally that
$$(\lambda y : T \,.\, (\lambda x : S \,.\, M))N = (\lambda x : S \,.\, (\lambda y : T \,.\, M)N).$$
8. (10 points) Some fun with combinators. Let
$$\begin{aligned} I &\stackrel{\text{def}}{=} \lambda x \,.\, x \\ K &\stackrel{\text{def}}{=} \lambda x \,.\, \lambda y \,.\, x \\ S &\stackrel{\text{def}}{=} \lambda x \,.\, \lambda y \,.\, \lambda z \,.\, xz(yz) \\ B &\stackrel{\text{def}}{=} S(KS)K. \end{aligned}$$
Using the equational rules, formally derive the normal forms (if any for the following):
   (a) $(SKK)x$
   (b) $Bxyz$
   (c) $(S(BBS)(KK))xyz$
9. (extra credit) Prove that application in the lambda calculus is not associative.
10. (extra credit only) Prove that every term of the simply-typed lambda calculus has a normal form.
11. (10 points) Using the encoding given Schmidt's exercise 13(c)(ii), called the Church numerals, give untyped lambda terms that implement:

(a) addition of church numerals, that is a lambda term, call it **plus** such that if e_n and e_m are lambda expressions that encode the numbers n and m, then (**plus** n m) is equivalent to the lambda expression that encodes the number $n + m$,

(b) multiplication, and

(c) (extra credit only) exponentiation.

12. (extra credit only) Suppose a symbol of the untyped λ-calculus alphabet is always 0.5 cm wide. Write down an untyped λ-calculus term with length less than 20 cm having a normal form with length at least $10^{10^{10}}$ light-year. (Note: the term must have a normal form.) The speed of light is $c = 3 \times 10^{10}$ cm/sec, so a light-year is about 1×10^{18} cm.

13. (20 points) Modify Paulson's lambda calculus interpreter to work for the simply typed lambda calculus. His code is found (on Zippy, Zaphod, and Flash) in

    `/usr/unsup/sml/lcp-book/Lambda.ML`

    and is described in chapter 9. The version of the simply typed lambda calculus you should use is one with only one type, call it `o`, and no constants.