Homework 9

Due Friday, 11/14/08

Please turn in your homework solutions by the beginning of class to the dropoff folder on `vpn.cs.pomona.edu` in directory `/common/cs/cs131/dropbox`. Make sure that all of your files include your name and the problem number in a comment.

1. (10 points) **Exceptions in ML**

   The function `stringToNum` defined below uses two auxiliary functions to convert a string of digits into a non-negative integer.

```
fun charToNum c = ord c - ord #"0";

fun calcList (nil,n) = n
  | calcList (fst::rest,n) =
                    calcList(rest,10 * n + charToNum fst);

fun stringToNum s = calcList(explode s, 0);
```

   For instance, `stringToNum "3405"` returns the integer 3405. (The function `explode` converts a string into a list of characters, and `ord` returns the ASCII integer value for a character.)

   Unfortunately, `calcList` returns a spurious result if the string contains any non-digits. For instance, `stringToNum "3a05"` returns 7905, while `stringToNum " 405"` returns ~15595. This occurs because `charToNum` will convert any character, not just digits. We can attempt to fix this by having `charToNum` raise an exception if it is applied to a non-digit.

   (a) Revise the definition of `charToNum` to raise an exception, and then modify the function `stringToNum` so that it handles the exception, returning ~1 if there is a non-digit in the string. You should make no changes to `calcList`.

   (b) Implement ML functions to provide the same behavior (including returning ~1 if the string includes a non-digit) as in the first part, but without using exceptions. While you may change any function, try to preserve as much of the structure of the original program as possible.

   (c) Which implementation do you prefer? Why?

   Turn in the code for parts (a) and (b) as files `except-a.ml` and `except-b.ml`.

2. (10 points) **Exceptions and Recursion**

   Please do problem 8.4 from Mitchell, page 230.

3. (5 points) **Tail Recursion**

   Please define a tail recursive function `sumsquares(n)` that can compute the sum of the first n squares: $1^2 + 2^2 + \ldots + n^2$.

4. (10 points) **Tail Recursion and Exception Handling**

   Please do problem 8.5 from Mitchell, page 230.

5. (10 points) **Continuations**

   Please do problem 8.9 from Mitchell, page 231.

6. (5 points) **Using Continuations in ML**

   For this problem I would like you to use the continuation library's `callcc` and `throw` available in the module `SMLofNJ.Cont`.

   The following program uses exceptions to short-circuit the calculation of the product of a list of integers if one of the elements is 0:

```
exception ZeroVal;

fun multListHelper [] = 1
  | multListHelper (fst::rest) = if fst = 0 then raise ZeroVal
   else fst*(multListHelper rest);

fun multExc list = (multListHelper list) handle ZeroVal => 0;
```

   `MultExc [2,3,4]` returns 24 as expected, but when `MultExc [1,2,3,4,5,6,7,0,8,9]` is evaluated, it short-circuits the multiplications and returns 0 as soon as it processes the 0 in the list.

   Please rewrite this function so that it uses continuations rather than exceptions. That is, rewrite `multExc` and `multListHelper` to use `callcc` and `throw` rather than exceptions. The overall structure of your program will be very similar.

   Don't forget to include `open SMLofNJ.Cont;` *at the beginning of your program!*