Reading

1. Mitchell, Chapter 8

Problems

1. (10 points) Exceptions

Consider the following functions, written in ML:

```ml
exception Excpt of int;
fun twice(f,x) = f(f(x)) handle Excpt(x) => x;
fun pred(x) = if x = 0 then raise Excpt(x) else x-1;
fun dumb(x) = raise Excpt(x);
fun smart(x) = (1 + pred(x)) handle Excpt(x) => 1;
```

What is the result of evaluating each of the following expressions?

(a) twice(pred,1);
(b) twice(dumb,1);
(c) twice(smart,0);

In each case, be sure to describe which exception gets raised and where.

2. (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.

```ml
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.

3. (10 points) ................................. Exceptions and Recursion

Mitchell, Problem 8.4

4. (5 points) ................................. Tail Recursion

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