Computer Science 15–212–X: Midterm Examination

October 3, 1996

Name

Name: _____

Andrew ID: _____

Instructions

- This is a closed-book, closed-notes examination.
- There are 10 pages in this examination, including two worksheets.
- This examination consists of 3 questions worth a total of 100 points. The point value of each question is given with the question.
- Read each question completely before attempting to solve any part.
- Write your answers legibly *in the space provided* on the examination sheet. If you use the back of a sheet, indicate clearly that you have done so on the front.
- Write your name and Andrew id in the space provided at the top of this page, and write your name at the top of each page in the space provided.
- Two worksheets are attached to the end of this examination for your personal use. *Under no circumstances will anything written on the worksheets count for or against your grade on this exam.*

Grading

Question	1	2	3	Total
Score				
Maximum	35	25	40	100

1 Induction and Recursion [35 points]

We define

```
type key = int
type 'a entry = key * 'a
datatype 'a tree = Empty | Node of 'a entry * 'a tree * 'a tree
```

as the type of binary search trees with integer keys and data of arbitrary type `'a`.

1. [5 pts] State the representation invariant of binary search trees.

2. [5 pts] Define a function `leaves` to count the number of `Empty` nodes in a binary search tree.

3. [10 pts] Let $|t|$ be the number of *entries* in a binary search tree t.
   Prove that `leaves`$(t) \hookrightarrow \overline{|t|+1}$, where $\overline{n}$ is the representation of n in ML.

4. [15 pts] Write a function

```
interval : int * int -> 'a tree -> 'a tree
```

where `interval` $(\overline{i}, \overline{j})$ t returns a binary search tree with those entries in t with a key strictly between i and j. You may assume without checking that $i < j$. Your function should exploit the representation invariant on t.

2 Abstract Types [25 points]

Consider a signature for positive rational numbers.

```
signature RATIONALS =
sig
  type rat
  exception Undefined
  val quotient : int * int -> rat
  val fraction : rat -> int * int
  val plus : rat * rat -> rat
  val equal : rat * rat -> bool
end;
```

We write $\overline{p}$ for the representation of an integer p in ML, and $\ulcorner \frac{p}{q} \urcorner$ for the representation of the rational $\frac{p}{q}$. This representation remains unspecified in the signature, but we require that a correct implementation of this signature satisfies the following properties.

- `quotient`$(\overline{p}, \overline{q}) \hookrightarrow \ulcorner \frac{p}{q} \urcorner$ if $p \geq 0$ and $q > 0$ and raises exception `Undefined` otherwise;
- `fraction`$(\ulcorner \frac{p}{q} \urcorner) \hookrightarrow (\overline{p'}, \overline{q'})$ where $\frac{p}{q} = \frac{p'}{q'}$ and $\frac{p'}{q'}$ is in cancelled form;
- `plus` and `equal` implement the obvious functions on positive rationals.

1. [15 pts] Implement a structure `R :> RATIONALS` where rationals are represented as pairs (numerator, denominator) in cancelled form. You may assume you have a function `gcd` defined by

```
fun gcd (0,n) = n
  | gcd (m,n) = gcd (n mod m, m);
```

2. After a correct declaration of the form `structure R :> RATIONALS = ...;`, what will be the results of the following expressions? In each case,

   - indicate either the type or a type error and explain the reason for the type error, and
   - if the expression is well-typed, give the value or the exception raised.

   (a) [2 pts] `R.fraction(R.quotient(15,6));`

   (b) [2 pts] `R.fraction(R.quotient(0,5)) handle R.Undefined => 0;`

   (c) [2 pts] `R.equal(R.quotient(2,3),R.quotient(~2,~3));`

   (d) [2 pts] `let open R in quotient(1,0) handle undefined => (0,1) end;`

   (e) [2 pts] `let open R in quotient((1,0) handle undefined => (0,1)) end;`

3 Higher-Order Functions [40 pts]

An infinite sequence $x_0, x_1, x_2, \ldots$ of real numbers can be represented by a function `x : int -> real` where $\texttt{x}(\overline{i}) \hookrightarrow \overline{x_i}$ for $i \geq 0$. This question explores this representation technique.

For each function you write below give its type, including auxiliary functions where necessary. You may always assume that the integer arguments to your functions are greater or equal to 0 without checking. Recall that the built-in function `real : int -> real` coerces an integer to the corresponding real number.

1. [5 pts] Give a representation of the sequence $1, \frac{1}{2}, \frac{1}{3}, \ldots$

2. [5 pts] Write a function `add` which, given a pair of sequences $x_0, x_1, \ldots$ and $y_0, y_1, \ldots$ computes the sequence $x_0 + y_0, x_1 + y_1, \ldots$.

3. [5 pts] Write a function `diff` which, given $x_0, x_1, x_2, \ldots$ computes the sequence of differences, consisting of $x_1 - x_0, x_2 - x_1, \ldots$.

4. [10 pts] Write a function `sum` which, given a sequence $x_0, x_1, x_2, \ldots$ returns the sequence consisting of the partial sums $x_0, x_0 + x_1, x_0 + x_1 + x_2, \ldots$.

5. [15 pts] Write a function `seqToList` which, given a sequence $x_0, x_1, \ldots$ and an integer $n \geq 0$ returns the list $[x_0, \ldots, x_{n-1}]$ of the first n elements of the sequence. In particular, `seqToList` s `0` should always return the empty list.

Worksheet

Worksheet