

BU CAS CS 320 (SUMMER I, 2009)
CONCEPTS OF PROGRAMMING LANGUAGES

Assignment 3

Due: June 8, 2009

Total: 150 points

Exercise 1 (30 points) See the on-line file assignment3.sats.

Exercise 2 (50 points) See the on-line file assignment3.sats.

Exercise 3 (70 points) Given a sequence s consisting of n elements a_1, \dots, a_n , we represent s as follows.

- If $n = 0$, that is, s is empty, we use **Empty** to represent s ;
- If $n = 2k$ for some $k > 0$, we use **Even** (\mathbf{xs}) to represent s , where \mathbf{xs} represents the following sequence of pairs;

$$(a_1, a_2), \dots, (a_{n-1}, a_n)$$

- If $n = 2k + 1$ for some $k \geq 0$, we use **Odd** (x , \mathbf{xs}) to represent s , where x represents a_1 and \mathbf{xs} represents the following sequence of pairs.

$$(a_2, a_3), \dots, (a_{n-1}, a_n)$$

We use the name random-access list for such a representation of l . The following datatype 'a **ralist** is declared for this kind of representation of list:

```
datatype pair (a:t@type) = S(a) of a | P(a) of (pair a, pair a)
datatype ralist (a:t@type) = Emp(a) of ()
                        | Evn(a) of ralist a
                        | Odd(a) of (pair a, ralist a)
```

Please implement the following operations on random-access lists. All the implementations should be of $O(\log n)$ -time complexity in order to receive full credit.

1. (10 points) Given an element x and a random-access list \mathbf{xs} , **racons** (x , \mathbf{xs}) generates a random-access list whose head and tail are x and \mathbf{xs} , respectively.
2. (10 points) Given a nonempty random-access list \mathbf{xs} , **rauncons** (\mathbf{xs}) return a pair consisting of the head and tail of \mathbf{xs} .
3. (20 points) Given a number n and a random-access list \mathbf{xs} , **ralookup** (n , \mathbf{xs}) returns the n th element (counting starts with 0) in \mathbf{xs} or issues an error if the length of the random-access list is less than or equal to n .

4. (30 points) Given a number n and a random-access list xs , `raupdate` (xs , n , x) updates the n th element (counting starts with 0) in xs with x or issues an error if the length of the random-access list is less than or equal to n . This one is a bit challenging, and the solution may involve the feature of higher-order function.

The types of these functions are given below:

```
fun{a:t@ype} racons (x: pair a, xs: ralist a): ralist a
fun{a:t@ype} rauncons (xs: ralist a): @(pair a, ralist a)
fun{a:t@ype} ralookup (xs: ralist a, i: int): pair a
fun{a:t@ype} raupdate (xs: ralist a, i: int, x: pair a): ralist a
```