CS 3721: Programming Languages Lab
Lab #06: ML programming using functions

The following are two equivalent function definitions in ML.

```
fun length(y) = case y of   nil => 0
                      |    x::xs => 1+length(xs);

fun length([]) = 0
   | length(x::y) = 1 + length(y);
```

Using either styles of programming, translate the following Scheme code to equivalent function definitions in ML.

1. ```
   (define find (lambda (x y)
                  (cond ((null? y) #f)
                        ((cons? y)
                         (if (eq? (car y) x) #t (find x (cdr y)))))))
   ```
   Use the following test cases for your ML code
   ```
   - find (3,[2,3,5]);
   val it = true : bool
   - find (3, [1,2,5]);
   val it = false : bool
   ```
2. ```
   (define Repeat (lambda (f n x)
                    (if (> n 1) (f (Repeat f (- n 1) x)) (f x))))
   ```
   Use the following test cases for your ML code
   ```
   - Repeat(fn x=>x+2, 3, 3);
   val it = 9 : int
   - Repeat(fn x=>x*5, 5, 5);
   val it = 15625 : int
   - Repeat (tl, 5, [1,2,3,4,5,6,7]);
   val it = [6,7] : int list
   ```
3. ```
   (define find (lambda (x y)
                  (cond ((null? y) #f)
                        ((number? y) (eq? x y))
                        ((cons? y)
                         (or (find x (car y)) (find x (cdr y)))))))
   ```
   In your translation, x should be treated as a single integer number, and the following datatype defines the type of y.
   ```
   datatype InputType = NULL | Number of int | Cons of InputType * InputType;
   ```
   Use the following test cases for your ML code
   ```
   - find(3, NULL);
   val it = false : bool
   - find (3, Number(4));
   val it = false : bool
   - find (3, Number(3));
   val it = true : bool
   - find (3, Cons(Number(5), Cons(Number(3), NULL)));
   ```
   ```

```
val it = true : bool
```

4. ```
   (define count (lambda (x y)
                   (cond ((null? y) 0)
                         ((number? y) (if (eq? x y) 1 0))
                         ((cons? y)
                          (+ (count x (car y)) (count x (cdr y)))))))
   ```
   Use the same types for x and y as those defined in the previous problem. Use the following test cases for your ML code

   ```
   - coount(3, NULL);
   val it = 0 : int
   - count (3, Number(4));
   val it = 0 : int
   - count (3, Number(3));
   val it = 1 : int
   - count (3, Cons(Number(2), Cons(Number(3), Cons(Number(3), NULL))));
   val it = 2 : int
   ```

After you are done, save your solution file and submit your file online at

`http://www.cs.utsa.edu/~cs3723`
   ```