CSE505 – Fall 2012
Assignment 3
Sample Solutions

1. Definition of lookup is as follows:

```
open Int;                (* this allows you to use the toString function *)

exception found of string * int;
exception notfound of string;

datatype treedict = leaf of string * int | node of treedict * treedict;

fun search(x, leaf(k,v))      =  if x=k  then  raise found(k,v)  else  false
   | search(x, node(t1,t2))  =  search(x,t1)  orelse  search(x,t2);

fun lookup(k,t)  =  let  val b = search(k,t)
                         in raise notfound(k)
                    end  handle found(k,v )   =>   "value of "  ^  k  ^  " is "  ^  toString(v)
                              | notfound(k)  =>   k  ^  " is not found";
```

2. Definitions of treemap and prune are as follows, assuming

```
datatype ‘a gametree = node of ‘a * ‘a gametree list;

fun treemap  f  node(p, glist)  =  node(f(p),  map2 (f, glist));
fun map2(f,  [ ])    = [ ]
   | map2(f,  n::t) =  (treemap f  n) :: map2(f,  t);

fun prune 0 node(p, _)      =  node(p,  nil)
   | prune n node(p, glist) =  node(p,  map prune(n-1) glist);
```

3. Consider the definition of the list of primes in a lazy functional language:

```
fun primes = sieve_all(numsfrom(2));

fun sieve_all( (thkp, thkt) ) =  let  fun thk() = sieve_all(sieve(thkt(), thkp()))
                                      in (thkp,  thk)
                                     end;

fun sieve ((thkh, thkt), p) = let fun thk() = sieve(thkt(), p)
                                   in  if  thkh() mod p = 0 then sieve(thkt(), p) else (thkh, thk)
                                  end;

fun numsfrom(n) = let fun thk1()  =  n
                          fun thk2()  =  numsfrom(n+1)
                      in (thk1, thk2)
                     end;
```

End of Sample Solutions