CSE505 – Fall 2012
Assignment 2- Sample Solution

1 [35%] Breadth-first iterator for binary tree:

```
class Tree {int val; Tree left; Tree right;}
class TreeList { Tree val; TreeList next; }

iterator int bf_elements(t) {
    TreeList tl := [t];
    while (tl != null) {
        Tree head = tl.val;
        tl := append(tl.next, [head.left, head.right]);
        yield head.val;
    }}
```

The notation [t_1 … t_n] refers to a list with elements t_1 ... t_n. The 'append' function was defined in the lecture slides.

2 [15%] Coroutine constructs cannot be translated in the same simple way that iterators are translated in terms of procedure parameters. The main problem lies in determining the body of the 'thunk' that would be passed as the actual parameter for the procedure parameter. In the case of iterators, the body of the foreach becomes the body of the thunk. Since coroutines need not be created or resumed in any specific context (such as a foreach), a systematic translation via procedure parameters is not easily achieved.

3 [50%] Assuming the following **stack** representation with `Lf.Lx.x` as the empty stack:

```
Lf.Lx.((f e1) ((f e2) … ((f en) x) …))
```

Following are non-recursive lambda-calculus definitions for the required stack operations.

```
let top = Ls.((s true) dummy)

let nonempty = Ls.((s Lx.Ly.true) false)

let size = Ls.Lf.Lx.((s Lx.f) x)

let push = Le.Ls.Lf.Lx.((f e) ((s f) x))
```

End of Sample Solutions