CS 3721: Programming Languages Lab
Lab #11: Continuation Passing

Example1:

```
fun length([]) = 0
  | length(y1::y2) = 1 + length(y2);
```

is converted to tail recursion using continuation passing in the following.

```
fun length2([],K) = K(0)
   | length2(y1::y2,K) = length2(y2, fn res => K(res+1));
```

Example2:

```
datatype tree = LEAF of int | NODE of tree*tree;
fun count(LEAF(y)) = 1
  | count(NODE(y1,y2)) = count(y1) + count(y2);
```

is converted to tail recursion in the following.

```
fun count2(LEAF(y),K) = K(1)
  | count2(NODE(y1,y2),K) = count2(y1,
                fn res1 => count2(y2, fn res2 => K(res1+res2)));
```

Translate the following ML code to tail recursion using continuation passing style.

1. ```
   fun find(x,[]) = 0
     | find(x,y1::y2) = if (x = y1) then 1+find(x,y2) else find(x,y2);
   ```
   Test your implementation with the following.

   ```
   find(3,[1,3,2],fn x=>x);
   find(3,[1,2,5,4],fn x=>x);
   ```

2. ```
   datatype tree = LEAF of int | NODE of tree*tree

   fun inTree(x, LEAF(y)) = x = y
     | inTree(x, NODE(y,z)) = inTree(x,y) orelse inTree(x,z);
   ```
   Test your implementation with the following.

   ```
   inTree(3, NODE(LEAF(5),NODE(LEAF(6),LEAF(3))), fn x=>x);
   inTree(3, NODE(LEAF(5),NODE(LEAF(6),LEAF(7))), fn x=>x);
   ```