```
********
Homework
********
```

- (easy) Finish the stack example by giving an implementation and proof outline for pop.
- (easy) Write out the proof outline for reverse in the functional version.
- (easy to medium) Specify and verify the following program copy takes a pointer to a binary tree, and copies it into a separate heap chunk, returning the pointer to the copy.

$$
\begin{array}{l}
\mathsf{copy}\ (p : \mathsf{ptr}) = \\
\quad \mathsf{if}\ p = \mathsf{null}\ \mathsf{then}\ \mathsf{returnnull} \\
\quad \mathsf{else}\ \ v \leftarrow p.value;\ tl \leftarrow p.left;\ tr \leftarrow p.right; \\
\qquad\quad p' \leftarrow \mathsf{alloc}\ v\ \mathsf{null}\ \mathsf{null}; \\
\qquad\quad tl' \leftarrow \mathsf{copy}\ tl;\ v'.left := tl'; \\
\qquad\quad tr' \leftarrow \mathsf{copy}\ tr;\ v'.right := tr'; \\
\qquad\quad \mathsf{return}\ p'
\end{array}
$$

- (difficult) Specify and verify a union-find data structure. It consists of a number of inverted trees. The nodes in a tree all have a parent field pointing to their parent.

  It exports the following methods.

  - find(x) returns a root of x, compressing the paths along the way.

$$
\begin{array}{l}
\mathsf{find}\ (x : \mathsf{ptr}) = \\
\quad i \leftarrow !(x.parent); \\
\quad \mathsf{if}\ i \neq \mathsf{null}\ \mathsf{then}\ j \leftarrow \mathsf{find}\ i;\ x.parent := j;\ \mathsf{return}\ j \\
\quad \mathsf{else}\ \mathsf{return}\ i
\end{array}
$$

  - union(x, y) joins the trees of x and y. In practice, structure keeps tree sizes, to join smaller to larger, but we ignore that here.

$$
\mathsf{union}\ (x\ y : \mathsf{ptr}) = i \leftarrow \mathsf{find}\ x;\ j \leftarrow \mathsf{find}\ y;\ \mathsf{if}\ i \neq j\ \mathsf{then}\ i.parent := j
$$