Only 1 and 6

## Ex 1

```rust
// RUST
fn memoize<U, V>(f: impl Fn(U) -> V) -> impl Fn(U) -> V
where
    U: Eq + Hash + Copy,
    V: Copy,
{
    let s: RefCell<HashMap<U, V>> = RefCell::new(HashMap::<U, V>::new());
    move |x: U| -> V {
        let mut h = s.borrow_mut();
        if !h.contains_key(&x) {
            let v = f(x);
            h.insert(x, f(x));
            v
        } else {
            *h.get(&x).unwrap()
        }
    }
}
```

## Ex 6

$$\begin{tikzcd}
 & \mathrm{Void} \arrow[loop right, "\mathrm{id}"] \arrow[dl, "\mathrm{absurd}"] \arrow[ddl, bend left, "\mathrm{absurd}"] \\
\mathrm{Bool} \arrow[loop right, "\mathrm{id}"] \arrow[loop right, distance=2em, "\mathrm{ff}"] \arrow[loop right, distance=3em, "\mathrm{ft}"] \arrow[loop right, distance=4em, "\mathrm{tf}"] \arrow[loop right, distance=5em, "\mathrm{tt}"] \arrow[d, bend right, "\mathrm{unit}"'] & \\
() \arrow[u, "\mathrm{t}"] \arrow[u, bend right, "\mathrm{f}"'] \arrow[loop right, "\mathrm{id}"] &
\end{tikzcd}$$

```haskell
{-# LANGUAGE LambdaCase #-}

import Data.Void

ff :: Bool -> Bool
ff = \case False -> False; True -> False

ft :: Bool -> Bool
ft = \case False -> False; True -> True

tf :: Bool -> Bool
tf = \case False -> True; True -> False

tt :: Bool -> Bool
tt = \case False -> True; True -> True

t :: a -> Bool
t _ = True
```

```
f :: a -> Bool
f _ = False

unit :: a -> ()
unit _ = ()
```