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

$$\begin{tikzcd}
\text{Void} \arrow[loop right, "id"] \arrow[d, bend right, "t"'] \arrow[d, "f"] \arrow[dd, bend left=60, "unit"] \\
\text{Bool} \arrow[loop right, "id"] \arrow[loop, out=20, in=-20, distance=3em, "ff"] \arrow[loop, out=15, in=-15, distance=5em, "ft"] \arrow[loop, out=10, in=-10, distance=7em, "tf"] \arrow[loop, out=5, in=-5, distance=9em, "tt"] \arrow[d, bend right, "unit"'] \\
() \arrow[loop right, "id"] \arrow[u, "t"] \arrow[u, bend right, "f"']
\end{tikzcd}$$

```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
```

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

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