Assosieringslister

En fattigmanns implementasjon av ordbok (avbildning, funksjon eller map) er såkalte assosieringslister. Det er lister på formen [(a,b)], hvor a sees på som en nøkkel og b som en verdi. En nøkkel kan finnes flere ganger, men da overskygger den fremste forekomsten alle de senere. For eksempel, hvis man slår opp i ordboken (tenkt som en liste av (nøkkel:verdi) par)

```
[("a",1),("b",2),("b",3)]
```

etter nøkkel b, får man verdien 2.

data Ordbok

Deklarer en type eller data `Ordbok n v`, med paramteretyper for `n`økler og verdier (og, dersom nødvendig, med passende konstruktorer). Er det noen krav man bør stille til paramtertyper? Vil man ha en tom ordbok?

finn :: Eq n => n -> Ordbok n v -> v

Programmer en slik funksjon som finner verdien til en nøkkel, f.eks.:

```
> finn "a" [("a",1),("b",2),("b",3)]
1
> finn "b" [("a",1),("b",2),("b",3)]
2
> finn "c" [("a",1),("b",2),("b",3)]
```

Her blir det vel et kall til Haskell `error` funksjonen som hever et unntak.

settInn :: Eq n => n -> v -> Ordbok n v -> Ordbok n v

Programmer en slik funksjon som setter inn paret `(a,b)::(n,v)` inn i `Ordbok n v`, slik at et oppslag etter nøkkelen `a` gir nå verdien `b`.

endre :: Eq n => n -> v -> Ordbok n v -> Ordbok n v

Programmer en slik funksjon som gitt en nøkkel `a::v`, en verdi `b::v` og en `Ordbok n v` endrer første tilfelle av `(a,_)` til `(a,b)` og returnerer den endrede Ordboken. Hvis `a` ikke finnes som nøkkel så returnes listen uendret.

slettF :: Eq n => n -> Ordbok n v -> Ordbok n v

Programmer en slik funksjon som gitt en nøkkel `a::n` og en `Ordbok n v` fjerner første tilfelle av `(a,_)` og returnerer den endrede listen. Hvis `a` ikke finnes som nøkkel så returnes listen uendret.

slettAlle :: Eq n => n -> Ordbok n v -> Ordbok n v

Skriv en funksjon som gitt en nøkkel `a::n` og en `Ordbok n v` fjerner alle tilfelle av `(a,_)` og returnerer den endrede listen. Hvis `a` ikke finnes som nøkkel så returnes listen uendret.

erLike :: Eq n => Eq v => Ordbok n v -> Ordbok n v -> Bool

Gitt at en `Ordbok n v` er en liste av par `[(n,v)]`, vil Haskell generere default tolkning av `==` for Ordbøker: to ordbøker er like hvis de er identiske som lister. Men et mer passende likhetsbegrep for ordbøker kan være at de er like hvis de har de samme nøklene og den samme verdien for hver nøkkel, f.eks. `[(‘‘a’’,1),(‘‘b’’,2)]` og `[(‘‘b’’,2),(‘‘a’’,1)]` er like ifølge denne definisjonen. Programmer en slik `erLike` funksjon.