5. Überblick: Functor, Monad, Applicative

Die Typklassen `Functor`, `Monad`, `Applicative` sind wie folgt definiert:

```
class Functor f where
    fmap :: (a -> b) -> f a -> f b
class (Functor f) => Applicative f where
    pure :: a -> f a
    (<*>) :: f (a -> b) -> f a -> f b
class (Applicative m) => Monad m where
    (>>=)  :: m a -> (a -> m b) -> m b
    return :: a -> m a
```

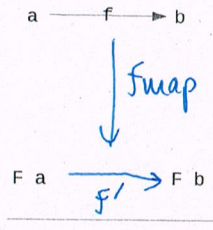
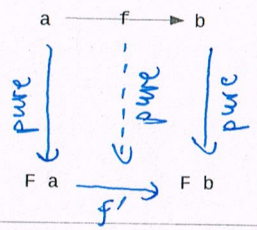
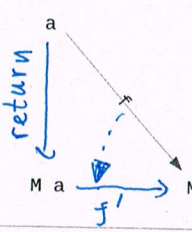
Zeichnen Sie in die folgenden kommutativen Diagramme Pfeile ein, welche die Funktionen / Operatoren

(i) `fmap` (`Functor`)

(ii) `pure`, `<*>` (`Applicative`)

(iii) `>>=`, `return` (`Monad`)

entweder direkt darstellen oder durch Zusatztext (wie eine ergänzende Gleichung) erläutern.

(i) Functor	(ii) Applicative	(iii) Monad
		
$f' = fmap\ f$	$f'\ x = (pure\ f) <*> x$	$f' = (=<<)\ f$ ↑ flipped bind $f'\ x = x >>= f$ bind

6. Springer-Züge auf dem Schachbrett

Im Schachspiel kann der Springer von seiner aktuellen Position aus in