head :: [a] \rightarrow a 
head (x:_) = x 
headV :: Vec (S n) a \rightarrow a 
headV (Vcons x _) = x 

\[ \text{partial} \quad \text{same implementation (modulo constructor names)} \]
\[ \text{but type is different} \]

\[ \uparrow \]

\[ \text{Undefined for empty list} \quad \text{defined for all values of} \]
\[ \text{type Vec (S n) a} \]

\[ \] 

\[ \text{data SNat} n \text{ where} \]
\[ \text{Zero :: SNat Z} \]
\[ \text{Succ :: SNat n \rightarrow SNat (S n)} \]
\[ \text{Test :: SNat True} \]

\[ \uparrow \alpha : \text{why does the type checker complain here and restrict the argument to Nat?} \]

A: after looking at the first constructor (Zero), it concludes that argument has to have kind Nat (only works if “data kinds” are enabled).
Second def. is ok, because \( n :: \text{Nat} \) and \( S n :: \text{Nat} \)
‘Test’ is not ok, because \( \text{True :: Bool} \)

Ok, but why is the compiler ok with this:

\[ \text{data Expr a where} \]
\[ \text{Add :: Expr Int \rightarrow Expr Int \rightarrow Expr Int} \]
\[ \text{BoolLit :: Expr Bool} \]

where \( \text{Expr has arg Int in first constructor, Bool in second.} \)

A: Because both Int and Bool have the same kind \( (\forall) \), just like \( \mathbb{Z}, n, S n \) in the SNat example have the same kind.