```
ouzo2:code$ ghci Reverse.hs
GHCi, version 8.2.2: http://www.haskell.org/ghc/  :? for help
Loaded GHCi configuration from /Users/hallgren/.ghci
[1 of 1] Compiling Main             ( Reverse.hs, interpreted )
Ok, one module loaded.
*Main> :r
[1 of 1] Compiling Main             ( Reverse.hs, interpreted )
Ok, one module loaded.
*Main> :set +s
*Main> sum [1..10000]
50005000
(0.01 secs, 2,404,152 bytes)
*Main> sum (reverse [1..10000])

<interactive>:4:6: error:
    • Variable not in scope: reverse :: [Integer] -> [Int]
    • Perhaps you meant ‘traverse’ (imported from Prelude)
(0.01 secs,)
*Main> sum (reverse_v1 [1..10000])
50005000
(1.25 secs, 4,454,452,232 bytes)
*Main> :r
[1 of 1] Compiling Main             ( Reverse.hs, interpreted )
Ok, one module loaded.
*Main> sum (reverse [1..10000])
50005000
(0.01 secs, 3,683,800 bytes)
*Main> rev
revOnto     reverse     reverse_v1
*Main> reverse [1..20]
[20,19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1]
(0.01 secs, 112,000 bytes)
*Main> sum (reverse [1..100000])
5000050000
(0.08 secs, 36,318,760 bytes)
*Main> sum (reverse [1..1000000])
500000500000
(0.83 secs, 362,536,928 bytes)
*Main> :l Queue.hs
[1 of 1] Compiling Queue            ( Queue.hs, interpreted )
Ok, one module loaded.
(0.03 secs,)
*Queue> empty
Q [] []
(0.00 secs, 69,800 bytes)
*Queue> add 1 it
Q [] [1]
(0.01 secs, 74,016 bytes)
*Queue> add 2 it
Q [] [2,1]
(0.00 secs, 74,216 bytes)
*Queue> add 3 it
Q [] [3,2,1]
(0.00 secs, 78,256 bytes)
```

```
*Queue> add 4 it
Q [] [4,3,2,1]
(0.00 secs, 78,664 bytes)
*Queue> q1=it
(0.00 secs, 0 bytes)
*Queue> q1
Q [] [4,3,2,1]
(0.00 secs, 78,536 bytes)
*Queue> isEmpty q1
False
(0.00 secs, 69,648 bytes)
*Queue> front q1
1
(0.00 secs, 65,792 bytes)
*Queue> remove q1
Q [2,3,4] []
(0.00 secs, 79,040 bytes)
*Queue> :r
[1 of 1] Compiling Queue            ( Queue.hs, interpreted )
Ok, one module loaded.
*Queue> empty
Q [] []
(0.00 secs, 73,808 bytes)
*Queue> add 1 it
Q [1] []
(0.01 secs, 73,960 bytes)
*Queue> add 2 it
Q [1] [2]
(0.00 secs, 74,344 bytes)
*Queue> add 3

<interactive>:26:1: error:
    • No instance for (Show (Q Integer -> Q Integer))
        arising from a use of ‘print’
        (maybe you haven't applied a function to enough arguments?)
    • In a stmt of an interactive GHCi command: print it
(0.00 secs,)
*Queue> add 3 it
Q [1] [3,2]
(0.00 secs, 74,656 bytes)
*Queue> add 4 it
Q [1] [4,3,2]
(0.00 secs, 78,912 bytes)
*Queue> q1=it
(0.00 secs, 0 bytes)
*Queue> front q1
1
(0.00 secs, 69,952 bytes)
*Queue> rem
rem     remove
*Queue> remove q1
Q [2,3,4] []
(0.00 secs, 79,200 bytes)
*Queue> q1==empty
```

```
<interactive>:32:1: error:
    • No instance for (Eq (Q Integer)) arising from a use of '=='
    • In the expression: q1 == empty
      In an equation for 'it': it = q1 == empty
(0.01 secs,)
*Queue> show q1==show empty
False
(0.01 secs, 75,520 bytes)
*Queue> :l Re
RecursiveDataTypes-ghci.pdf  RecursiveDataTypes-ghci.txt
Reverse.hs
*Queue> :l Reverse.hs
[1 of 1] Compiling Main             ( Reverse.hs, interpreted )

Reverse.hs:17:16: error:
    Ambiguous occurrence 'sum'
    It could refer to either 'Prelude.sum',
                             imported from 'Prelude' at Reverse.hs:12:1-14
                             (and originally defined in 'Data.Foldable')
                          or 'Main.sum', defined at Reverse.hs:16:1
   |
17 | sum (x:xs) = x+sum xs
   |                ^^^
Failed, no modules loaded.
(0.00 secs,)
Prelude> :r
[1 of 1] Compiling Main             ( Reverse.hs, interpreted )
Ok, one module loaded.
*Main> reverse []

<interactive>:36:1: error:
    Ambiguous occurrence 'reverse'
    It could refer to either 'Prelude.reverse',
                             imported from 'Prelude' at Reverse.hs:12:1-14
                             (and originally defined in 'GHC.List')
                          or 'Main.reverse', defined at Reverse.hs:27:1
(0.00 secs,)
*Main> :r
[1 of 1] Compiling Main             ( Reverse.hs, interpreted )
Ok, one module loaded.
*Main> reverse []
[]
(0.00 secs, 68,960 bytes)
*Main> :i Num
class Num a where
  (+) :: a -> a -> a
  (-) :: a -> a -> a
  (*) :: a -> a -> a
  negate :: a -> a
  abs :: a -> a
  signum :: a -> a
  fromInteger :: Integer -> a
  {-# MINIMAL (+), (*), abs, signum, fromInteger, (negate | (-)) #-}
```

```
        -- Defined in ‘GHC.Num’
instance Num Word -- Defined in ‘GHC.Num’
instance Num Integer -- Defined in ‘GHC.Num’
instance Num Int -- Defined in ‘GHC.Num’
instance Num Float -- Defined in ‘GHC.Float’
instance Num Double -- Defined in ‘GHC.Float’
*Main> fm x y = x == y+1
(0.00 secs, 0 bytes)
*Main> :t fm
fm :: (Eq a, Num a) => a -> a -> Bool
*Main> :l Set.hs
[1 of 1] Compiling Set              ( Set.hs, interpreted )
Ok, one module loaded.
(0.01 secs,)
*Set> filter even [1..10]
[2,4,6,8,10]
(0.01 secs, 77,072 bytes)
*Set> s1 = Set even
(0.00 secs, 0 bytes)
*Set> member 5 s1
False
(0.01 secs, 72,760 bytes)
*Set> member 4 s1
True
(0.01 secs, 68,784 bytes)
*Set> s2 = comp
compare     complement
*Set> s2 = complement s1
(0.00 secs, 0 bytes)
*Set> member 5 s2
True
(0.01 secs, 73,120 bytes)
*Set> member 4 s2
False
(0.01 secs, 72,872 bytes)
*Set> s3 = Set odd
(0.00 secs, 0 bytes)
*Set> mem
member  mempty
*Set> member 5 s3
True
(0.01 secs, 72,912 bytes)
*Set>
Leaving GHCi.
ouzo2:code$
```