```
scala> def sum (xs : List[Int]) : Int =
             xs match {
                case Nil => 0
                case x :: ys => x + sum(ys)
             }
sum: (xs: List[Int])Int
scala> sum (List(1,2,3))
res2: Int = 6
```

- Remark: In Scala, recursive function definitions require explicit *result types* (Int in the above) as well as parameter types (List[Int]), which are always required.

(*)

Polymorphic functions

- In Haskell:

  length :: [a] → Int

- Type parameters in Scala (T in the example below)

```
scala>  def length [T] (xs: List[T]) : Int =
            xs match {
                case Nil => 0
                case x :: ys => length(ys) + 1
            }
length: [T](xs: List[T])Int
scala> length (List(1,2,3))
res3: Int = 3
```

(*)