```
In [13]: import $file.hw2stdlib_new
         import hw2stdlib_new._
```

```
Compiling hw2stdlib_new.sc
```

```
Out[13]: import $file.$

         import hw2stdlib_new._
```

Homework 2

Due 9/12 at 11:59pm

Submission Instructions

Upload only this .ipynb file to Canvas. Do not add anything to hw2stdlib since you can't submit it.

In this homework we will develop more useful functions for numbers as well as some of the functions on lists that we spoke about during lecture.

Arithmetic Functions

Problem 1 (5 Points)

Write the minus function, which subtracts a natural number from another natural number. If the result would be less than zero then leave the answer as zero.

```
In [14]: def minus(x : Nat, y : Nat) : Nat =
             // BEGIN SOLUTION
             (x, y) match {
                 case (Zero, _)           => Zero
                 case (x, Zero)           => x
                 case (Succ(x), Succ(y)) => minus(x, y)
             }
             // END SOLUTION
```

```
Out[14]: defined function minus
```

```
In [19]: assert(minus(four, two) == two, 1)
         assert(minus(four, Zero) == four, 2)
         assert(minus(Zero, four) == Zero, 3)
         passed(3)
```

```
*** Tests Passed (3 points) ***
```

```
In [16]: // HIDDEN TEST (2 pts)
         // BEGIN HIDDEN TESTS
         assert(minus(Zero, Zero) == Zero, 1)
         // END HIDDEN TESTS
```

Problem 2 (5 points)

Define the `pow` function, which raises a natural number to some power. Notationally:

$$f\ x\ y = x^y$$

Hint: Use the mult function from the stdlib for help

```
In [28]: // BEGIN SOLUTION
         def pow(x : Nat, y : Nat) : Nat = y match {
             case Zero       => one
             case Succ(Zero) => x
             case Succ(yy)   => mult(x, pow(x, yy))
         }
         // END SOLUTION
```

```
Out[28]: defined function pow
```

```
In [31]: assert(pow(three, two) == nine, 1)
         assert(pow(four, Zero) == one, 2)
         assert(pow(two, two) == four, 3)
         passed(4)
```

```
*** Tests Passed (3 points) ***
```

```
In [30]: // HIDDEN TEST (1 pt)
         // BEGIN HIDDEN TESTS
         assert(pow(two, one) == two, 3)
         // END HIDDEN TESTS
```

Problem 3 (5 points)

Write the "equals to" function for numbers. This function should return $True$ if $x == y$ and $False$ otherwise.

```
In [33]: def eq_nat(x : Nat, y : Nat) : Bool =
             // BEGIN SOLUTION
             (x, y) match {
                 case (Zero, Zero)       => True
                 case (Succ(x), Succ(y)) => eq_nat(x, y)
                 case (x, y)             => False
             }
             // END SOLUTION
```

```
Out[33]: defined function eq_nat
```

In [35]:
```
assert(eq_nat(four, four) == True, 1)
assert(eq_nat(four, Zero) == False, 2)
assert(eq_nat(eight, nine) == False, 3)
assert(eq_nat(nine, nine) == True, 4)
passed(4)
```

```
*** Tests Passed (3 points) ***
```

In [36]:
```
// HIDDEN TEST (1 pt)
// BEGIN HIDDEN TESTS
assert(eq_nat(Zero, Zero) == True, 1)
// END HIDDEN TESTS
```

Functions on Number Lists

Problem 4 (5 pts)

Write the sum function. This function should take a list of natural numbers and return their sum. Here is the type:

$$sum : \text{List}\,\mathbb{N} \to \mathbb{N}$$

Hint: An empty list should have sum 0.

In [37]:
```
// BEGIN SOLUTION
def sum(xs : List[Nat]) : Nat = xs match {
    case Empty       => Zero
    case Cons(x, xs) => plus(x, sum(xs))
}
// END SOLUTION
```

Out[37]:
```
defined function sum
```

In [39]:
```
assert(sum(Cons(one, Empty)) == one, 1)
assert(sum(Cons(one, Cons(seven, Cons(two, Empty)))) == ten, 2)
passed(4)
```

```
*** Tests Passed (4 points) ***
```

In [40]:
```
// HIDDEN TEST (1 pt)
// BEGIN HIDDEN TESTS
assert(sum(Empty) == Zero)
// END HIDDEN TESTS
```

Problem 5 (5 pts)

Write the product function. This function should take alist of natural numbers and return their product. Here is the type:

$$prod : \text{List } \mathbb{N} \to \mathbb{N}$$

Hint: An empty list should have product 1.

```
In [43]: // BEGIN SOLUTION
         def prod(xs : List[Nat]) : Nat = xs match {
             case Empty       => one
             case Cons(x, xs) => mult(x, prod(xs))
         }
         // END SOLUTION
```

```
Out[43]: defined function prod
```

```
In [44]: assert(prod(Cons(one, Empty)) == one, 1)
         val fourteen = Succ(Succ(Succ(Succ(ten))))
         assert(prod(Cons(one, Cons(seven, Cons(two, Empty)))) == fourteen, 2)
         passed(4)
```

```
*** Tests Passed (4 points) ***
```

```
Out[44]: fourteen: Succ = Succ(Succ(Succ(Succ(Succ(Succ(Succ(Succ(Succ(Succ(Succ
         (Succ(Succ(Succ(Zero))))))))))))))
```

```
In [47]: // HIDDEN TEST (1 pt)
         // BEGIN HIDDEN TESTS
         assert(prod(Empty) == one)
         // END HIDDEN TESTS
```

Problem 6 (6 pts)

Write the function that tells us if a given natural number is in a list of natural numbers. (you will need to use the equals function defined in problem 3)

$$in : \mathbb{N} \to \text{List } \mathbb{N} \to \mathbb{B}$$

```
In [48]: // BEGIN SOLUTION
         def in(n : Nat, xs : List[Nat]) : Bool = xs match {
             case Empty       => False
             case Cons(x, xs) => eq_nat(x, n) match {
                 case True  => True
                 case False => in(n, xs)
             }
         }
         // END SOLUTION
```

```
Out[48]: defined function in
```

```
In [49]: assert(in(one, Cons(one, Empty)) == True, 1)
         assert(in(seven, Cons(one, Cons(seven, Cons(two, Empty)))) == True, 2)
         assert(in(eight, Cons(one, Cons(seven, Cons(two, Empty)))) == False, 3)
         assert(in(seven, Empty) == False, 4)
         passed(6)
```

```
*** Tests Passed (6 points) ***
```