1. xs.map(id) = xs

```
Base case: xs =[]

Prove: [].map(id) = []

      LHS   = [].map(id)
            = []                                       (map definition)
      RHS:  = [] = LHS

Induction case: xs = x:xs’

IH: Assume xs’.map(id) = xs’ is true for any xs’.

      LHS   = (x:xs').map(id)
            =  x:xs'                                   (map definition)
      RHS   =  x:xs'
      LHS   = RHS

Done
```

2. (xs.append(ys)).map(f) = (xs.map(f)).append(ys.map(f))

```
Base case: xs =[]

Prove: ([].append(ys)).map(f) = ([].map(f)).append(ys.map(f))

      LHS   = ([].append(ys)).map(f)
            = ys                              (append, map definition)
      RHS   = ([].map(f).append(ys.map(f))
            = [].append(ys.map(f))
            = [].append(ys)                   (append, map definition)
            = ys = LHS

Induction case: xs = x:xs’

IH: assume (xs’.append(ys)).map(f) = (xs’.map(f)).append(ys.map(f)) is
true for any xs’.

      LHS   = ((x:xs’).append(ys)).map(f)
            = x:(xs’.append(ys)).map(f)
            = x:(xs’.append(ys).map(f))       (append, map definition)
      RHS   = ((x:xs’).map(f)).append(ys.map(f))
            = x:(xs.append(ys).map(f))        (append, map definition)
      LHS   = RHS

Done
```

3. (xs.append(ys)).fold(g, a) = xs.fold(g, ys.fold(g, a))

Base case: xs =[]

Prove: ([].append(ys)).fold(g, a) = [].fold(g, ys.fold(g, a))

```
LHS   = ([].append(ys)).fold(g, a)           (append  definition)
      = ys .fold(g, a)                          (fold definition)
RHS   = [].fold(g, ys.fold(g, a))
      = ys.fold(g, a)                           (fold definition)
      = ys = LHS
```

Induction case: xs = x:xs’

IH: assume (xs’.append(ys)).map(f) = (xs’.map(f)).append(ys.map(f)) is true for any xy.

```
LHS   = ((x:xs’).append(ys)).map(f)
      = x:(xs’.append(ys)).map(f)
      = x:(xs’.append(ys).map(f))        (append, map definition)
RHS   = ((x:xs’).map(f)).append(ys.map(f))
      = x:(xs’.append(ys).map(f))        (append, map definition)
LHS   = RHS
```

Done

4. (xs.append(ys)).length() = xs.length() + ys.length()

Base case: xs =[]

Prove: ([].append(ys)).length() = [].length() + ys.length()

```
LHS   = ([].append(ys)).length()
      = ys.length()                   (append, length definition)
RHS   = [].length() + ys.length()
      = ys.length() = LHS                     (length definition)
```

Induction: xs = x:xs’

IH: Assume (xs’.append(ys)).length() = xs’.length() + ys.length() is true for any xy.

```
LHS   = (x:xs’).append(ys)).length()
      = x:(xs’.append(ys).length())   (append, length definition)
      = 1 + xs’.length() + ys.length()
RHS   = (x:xs’).length() + ys.length()
      = x:(xs’.append(ys))+ys.length()(append, length definition)
      = 1 + xs’.length() + ys.length()
LHS   = RHS
```

Done

5. (xs.reverseH (ys)).length() = xs.length() + ys.length()

```
Base case: xs =[]

Prove: ([].reverseH(ys)).length() = [].length() + ys.length()

      LHS   = ([].reverseH(ys)).length()

            = ys.length()                 (reverseH, length definition)

      RHS   = [].length() + ys.length()

            = ys.length() = LHS                     (length definition)

Induction: xs = x:xs’

IH: Assume(xy.reverseH (ys)).length() = xy.length() + ys.length() is
true for any xy.

      LHS   = ((x:xs’).reverseH(ys)).length()

            = xs’.length(x:ys).length()            (reveseH definition)

            = xs’.length() + (x:ys).length()

            = 1 + xs’.length() + ys.length()        (length definition)

      RHS   = (x:xs’).length() + ys.length()

            = 1 + xs’.length() + ys.length()        (length definition)

      LHS   = RHS

Done
```

6. xs.length() = (xs.reverse2 ()).length()

```
Prove: xs.length() = (xs.reverse2()).length()

      LHS   = xs.length()


      RHS   = (xs.reverse2()).length()

            = xs.reverseH([]).length()             (revese2 definition)

            = xs.length() + [].length()            (reveseH definition)

            = xs.length() = LHS                     (length definition)

      LHS   = RHS

Done
```

7. (xs.append(ys)).reverse() = ys.reverse().append(xs.reverse())

Base case: `xs =[]`

Prove: `([].append(ys)).reverse() = ys.reverse().append([].reverse())`

```
LHS   = ([].append(ys)).reverse()
      = ys.reverse()                  (append, reverse definition)
RHS   = ys.reverse().append([].reverse())
      = ys.reverse()                  (append, reverse definition)
```

Induction: `xs = x:xs’`

IH: `Assume(xy.append(ys)).reverse() = ys.reverse().append(xy.reverse()) is true for any xy.`

```
LHS   = (x:xs’.append(ys)).reverse()
      = x:(xs’append(ys).reverse())            (append definition)
      = xs’.append(ys).reverse().append(x:[])
      = ys.reverse().append(xs’.reverse()).append(x:[])


RHS   = ys.reverse().append(x:xs’.reverse()) (reverse definition)
      = ys.reverse().append(xs’.reverse()).append(x:[])
LHS   = RHS
```

Done

8. (xs.reverse()).reverse() = xs

Base case: `xs =[]`

Prove: `([].reverse()).reverse() = []`

```
LHS   = ([].reverse()).reverse()
      = [].reverse()                           (reverse definition)
RHS   = []
```

Induction: `xs = x:xs’`

IH: `Assume (xs’.reverse()).reverse() = xs’ is true for any xs’.`

```
LHS   = (x:xs’.reverse()).reverse()
      = x:(xs’.reverse().reverse())          (reverse definition)
      = x:xs’
RHS   = x:xs’


LHS   = RHS
```

Done

9. xs.reverseH (ys) = (xs.reverse()).append(ys)

```
Base case: xs =[]

Prove: [].reverseH (ys) = ([].reverse()).append(ys)

    LHS  = [].reverseH (ys)                    (reveseH definition)
         = ys
    RHS  = ([].reverse()).append(ys)
         = [].append(ys)               (append, reverse definition)
         = ys

Induction: xs = x:xs’

IH: Assume xs’.reverseH(ys) = (x:xs'.reverse()).append(ys) is true for
any xs’.

    LHS  = x:xs’.reverseH(ys)
         = xs’.reverseH(x:ys)                 (reverseH definition)
         = (x:xs').reverse()).append(ys)        (append definition)
    RHS  = (x:xs'.reverse()).append(ys)
         = (xs’.reverse()).append(ys)          (reverse definition)
    LHS  = RHS

Done
```

10. (xs.reverseH (ys)).reverseH (zs) = ys.reverseH (xs.append(zs))

```
Base case: xs =[]

Prove: ([].reverseH(ys)).reverseH(zs) = ys.reverseH([].append(zs))

    LHS  = ([].reverseH(ys)).reverseH (zs)
         = ys.reverseH(zs)                    (reverseH definition)
    RHS  = ys.reverseH([].append(zs))
         = ys.reverseH(zs)                      (append definition)

Induction: xs = x:xs’

IH: Assume(xs'.reverseH(ys)).reverseH(zs) = ys.reverseH(xs'.append(zs))
is true for any xs’.

    LHS  = ((x:xs').reverseH (ys)).reverseH (zs)
         =  ys.reverseH((x:xs').append(zs))   (reverseH definition)
    RHS  = ys.reverseH(xs'.append(zs))
    LHS  = RHS
```

Done

11. xs.reverseH (ys.append(zs)) = (xs.reverseH (ys)).append(zs)

```
Prove: xs.reverseH(ys.append(zs)) = xs.reverseH(ys).append(zs)


      LHS   = (xs.reverse()).append(ys.append(zs))

            = ((xs.reverse().append(ys)).append(zs)(append association)

            = (xs.reverseH(ys)).append(zs)

            = RHS


      LHS   = RHS

Done
```

12. (xs.append(ys)).reverseH (zs) = ys.reverseH (xs.reverseH (zs))

```
Base case: xs =[]

Prove: ([].append(ys)).reverseH(zs) = ys.reverseH([].reverseH(zs))

      LHS   = ([].append(ys)).reverseH(zs))

            = ys.reverseH(zs)                        (append definition)


      RHS   = ys.reverseH([].reverseH (zs))

            = ys.reverseH(zs)                      (reverseH definition)

Induction: xs = x:xs’

IH: Assume(xs'.append(ys)).reverseH(zs)=ys.reverseH(xs'.reverseH (zs))
is true for any xs’.

      LHS   = ((x:xs').append(ys)).reverseH(zs)

            = (xs.append(ys).reverse()).append(zs)          (solution 9)

            = (ys.reverse().append(xs.reverse())).append(zs)(solution 7)

            = ys.reverseH(xs.reverse().append(zs))         (solution 9)

            = ys.reverseH ((x:xs').reverseH (zs))  (reverseH definition)


      RHS   = ys.reverseH(xs.reverseH(zs))

      LHS   = RHS

Done
```

13. (xs.append(ys)).reverse2 () = ys.reverse2 ().append(xs.reverse2 ())

Base case: xs=[]

Prove:([].append(ys)).reverse2() = ys.reverse2().append([].reverse2())

```
LHS   =([].append(ys)).reverse2()
      = ys.reverse2()
RHS   =ys.reverse2().append([].reverse2())
      = ys.reverse2().append([])
      = ys.reverse2()
```

Indective case: xs=x:xs'

IH:
Assume(xs'.append(ys)).reverse2()=ys.reverse2().append(xs'.reverse2())
is true for any xs'.

```
LHS   = (x:xs.append(ys)).reverse2()
      = (x:xs.append(ys)).reverseH([])
      = (xs.append(ys).reverseH(x:[])
      = ys.reverseH(xs.reverseH(x:[]))               (solution 12)

RHS   = ys.reverse2().append((x:xs).reverse2())
      = ys.reverseH([]).append(xs.reverseH(x:[]))       (reverse2)
        = ys.reverseH(xs.reverseH(x:[]))             (solution 11)
LHS   = RHS
```

Done

14. (xs.reverse2 ()).reverse2 () = xs

```
Base case: xs=[]

Prove: ([].reverse2()).reverse2() = []

      LHS   =([].reverse2()).reverse2()

            = [].reverseH([]).reverseH([])

            = []

      RHS   = []

Inductive case: xs=(x:xs’)

IH: (xs’.reverse2()).reverse2() = xs’ is true for any xs’.


      LHS   =((x:xs).reverse2()).reverse2()

            =((x:xs).reverseH([])).reverseH([])   (reverse2 definition)

            = xs.reverseH(x:[]).reverseH([])

            = (x:[]).reverseH(xs.append([]))              (solution 10)

            = x.reverseH(xs)

            = x.reverse().append(xs)                        (solution 9)

            = x.append(xs’)


      RHS   = x:xs’

            = x.append(xs’)

      LHS   = RHS
```

Done

15. t.flattenH (xs) = t.flatten().append(xs)

```
Base case: t = d
      LHS   = d.flattenH(xs) = d:xs

      RHS   = d.flatten().append(xs)
            = (d:[]).append(xs)
            = d:[].append(xs)
            = LHS
Inductive case: t = N(d,t1,t2)
      xs.append(ys.append(zs)) = (xs.append(ys)).append(zs)
prove: N(d,t1,t2).flattenH(xs) = N(d,t1,t2).flatten().append(xs)
      IH1:  t1.flattenH(xs) = t1.flatten().append(xs)
      IH2:  t2.flattenH(xs) = t2.flatten().append(xs)

      LHS   = N(d,t1,t2).flattenH(xs)
            = t1.flattenH(d:t2.flattenH(xs))
            = t1.flatten().append(d:t2.flattenH(xs))
            = t1.flatten().append(d:t2.flatten().append(xs))

      RHS   = N(d,t1,t2).flatten().append(xs)
            = (t1.flatten().append(d:t2.flatten())).append(xs)
            = t1.flatten().append(d:t2.flatten().append(xs))
      LHS   = RHS
Done
```

16. t.flatten2 () = t.flatten()

```
Prove:
      LHS   = t.flatten2()                       (flatten2 definition)
            = t.flattenH([])                     (flattenH definition)
            = t.flatten().append([])               (append definition)
            = t.flatten()
      LHS   = RHS
Done
```

17. t.map(f1).sum() = t.nodes()

```
Base case: t = leaf(d)
Prove: leaf(d).map(f1).sum()=leaf(d).nodes()
      LHS   = leaf(d).map(f1).sum()
            = leaf(f1.apply(d)).sum()              (definition of map)
            = leaf(1).sum()                              (f1 definition)
            = 1
      RHS   = leaf(d).nodes()
            = 1
Induction case: T= N(d,t1,t2)
      Assume t1.map(f1).sum() = t1.nodes()
      Assume T2. map(f1).sum() = t2.nodes()

      LHS   = N(d,t1,t2). map(f1).sum()
            = N(f1.apply(d),t1.map(f), t2.map(f)).sum()
            = 1 + (t1.map(f), t2.map(f)).sum()           (f1 definition)
            = 1 + t1.map(f).sum() + t2.map(f).sum()
            = 1 + t1.nodes() + t2.nodes()                (IH Assumption)
            = N(d,t1,t2).nodes()

      RHS   = N(d,t1,t2).nodes()
      LHS   = RHS
Done
```

18. t.nodes() = t.longestPath().length() + 1.

Prove:

```
Base case: t = leaf(d)

      LHS   = leaf(d).nodes()

            = 1

      RHS   = leaf(d).longestPath().length()+1

            = 1 + 1                             (longestPath definition)

            = 2

      LHS   not equal to RHS

Not a statement

Done.
```

19. For non-empty trees t, it is the case that t.internalNodes() + 1 = t.leaves().

```
Base case:

        Nodes = 1

        Leaf  = d

        Internal nodes = 0

        LHS   = leaf(d).internalNodes() + 1

              = 0 + 1

              = 1

        RHS   = leaf(d).leaves()

              = 1

Induction case: T= N(d,t1,t2)

        Assume t1.internalNodes() + 1 = t1.leaves()

        Assume t2.internalNodes() + 1 = t2.leaves()


        LHS   = N(d,t1,t2).internalNodes() + 1

              = leaf(d).internalNodes()+ N(t1,t2).internalNodes() + 1

              = t1.internalNodes() + t2.internalNodes() + 1

              (IH Assumption)

              = t1.internalNodes() + t2.leaves()


        RHS   = N(d,t1,t2).leaves()

              = t1.leaves + t2.leaves()

              = t1.internalNodes() + 1 + + t2.leaves()

              (IH Assumption)

        LHS   not equal to RHS

Not a statement

Done.
```

20. A full m-ary with n nodes has (n − 1)/m internal nodes and ((m − 1)n + 1)/m leaves.

```
Base case:
    Nodes = 1
    Leaf  = d
    Internal nodes = 0
    (n - 1)/m    = (1-1)/m
                 = 0
    ((m - 1)n + 1)/m  = ((m-1)(1)+1)/m
                      = (m-1+1)/m
                      = 1
Induction case: T= N(d,t1,t2, ..., tm)
    LHS   = i1 +i2 + ... + im + 1
          = (n1-1)/m + (n2-1)/m + ...+ (nm-1)/m + 1
          = (n1-1 + n2-1 + ... + nm-1)/m + m/m
```

21. A full m-ary with i internal nodes has mi + 1 nodes and (m − 1)i + 1 leaves.

22. A full m-ary with l leaves has (ml − 1)/(m − 1) nodes and (l − 1)/(m − 1) internal nodes.

23. How many people have seen the letter, including the first person?

24. How many people sent out the letter?