IMPERATIVE PROGRAMMING 2017 MOISE GABRIEL

QUESTION 4

(a)

```
def reduce (m : int, n : int) : (int, int) = {
    var a = m
    var b = n
    // m/n can be simplified into a fraction which is in "its lowest terms" if we simplify
it by gcd (m, n), which we will calculate below
    // invariant i : gcd (m, n) = gcd (a, b)
    while (b != 0)
    { var r = a % b
      a = b
      b = r
      // Here, we use the fact that gcd (a, b) = gcd (b, a % b)
    }
    // b = 0 => gcd (m, n) = a
    return (m/a, n/a)
}
```

(b) We want to calculate the biggest positive integer q such that $\frac{1}{q} \leq \frac{m}{n}$. As $\lceil \frac{n}{m} \rceil - 1 < \frac{n}{m} \leq \lceil \frac{n}{m} \rceil \Rightarrow \frac{1}{\lceil \frac{n}{m} \rceil - 1} > \frac{m}{n} \geq \frac{1}{\lceil \frac{n}{m} \rceil} \Rightarrow q = \lceil \frac{m}{m} \rceil$

The algorithm we will use is:

1. calculate $\lceil \frac{n}{m} \rceil$
2. replace $\frac{m}{n}$ with $\frac{m}{n} - \frac{1}{\lceil \frac{n}{m} \rceil}$
3. repeat 1, 2 until m becomes 0

The algorithm will terminate because $\frac{m}{n} - \frac{1}{q} = \frac{mq - n}{nq}$ and $mq - n = m \lceil \frac{n}{m} \rceil - n <$ $< m\left(\frac{n}{m} + 1\right) - n = \cancel{n} + m - \cancel{n} = m \Rightarrow$ the new m is going to be smaller than m, so, after at most m step we will get to $m = 0$ and the algorithm will terminate.

(c)

```
def fromRat (m1 : int, n1 : int) : Array [Int] = {
    var m = m1; var n = n1
    var e = new Array [Int] (1000)
    var k = 0
    // invariant j : $\frac{m_1}{n_1} = \frac{m}{n} + \sum_{i=0}^{k-1} \frac{1}{e(i)}$
    while (m!=0)
    { var q = n/m
      if (n % m !=0) q+=1
      // $q = \lceil \frac{n}{m} \rceil$
      e(k) = q ; k+=1
      m = m*q-n ; n = n*q
    }
    // $m=0 \Rightarrow \frac{m_1}{n_1} = \sum_{i=0}^{k-1} \frac{1}{e(i)}$
    e
}
```

(d)

```
def toRat(e : Array[Int]) : (Int, Int) = {
    var N = e.size
    var m = 0 ; var n = 1
    var i = 0
    // invariant i : $\frac{m}{n} = \sum_{j=0}^{i-1} \frac{1}{e(i)}$
    while (i < N)
    { m = m * e(i) + n
      n = n * e(i)
      i += 1
    }
    reduce(m,n)
}
```

(e)

```
// Printing a fraction
def printRat(m: Int, n: Int): String = m.toString + "/" + n.toString

// Printing the Egyptian fractions
def printEgypt(e : Array[Int]) : String = {
    var N = e.size
    while (e(N-1) == 0) N -= 1   // when the size of the array is bigger than the number
                                    of elements in it
    var str = ""
    for (i <- 0 until N-1) str = str + "1/" + e(i).toString + "+"
    str = str + "1/" + e(N-1)
    str }
```

```
// if we want to print an equality :
def printEq (m: int, n: int) : String ={
    printRat(m,n) + " = " + printEgypt (fromRat(m,n))
}
```