Exercises 3

1 Loop semantics

Compute and simplify the relation corresponding to the following programs:

```
y = 0
while (z > 0) {
  y = y + x
  z = z − 1
}
```

```
c = 0
while (b >= a) {
  b = b − a
  c = c + 1
}
```

2 Loop invariants

In the following program, provide the necessary loop invariant for verification to succeed:

```
def binarySearch(a: Array[BigInt], key: BigInt): Int = ({
  require(a.length > 0 && forall { (i: Int, j: Int) =>
    (i >= 0 && j >= 0 && i < a.length && j < a.length && i < j) ==> (a(i) <= a(j))
  })

  var low = 0
  var high = a.length − 1
  var res = −1

  (while(low <= high && res == −1) {
    val o = if ((high & 1) == 1 && (low & 1) == 1) 1 else 0
    val i = high / 2 + low / 2 + o
    val v = a(i)

    if(v == key)
      res = i

    if(v > key)
      high = i − 1
    else if(v < key)
      low = i + 1
  }) invariant(TODO)
  res
}) ensuring(res => {
  if(res == −1)
    forall((i: Int) => (0 <= i && i < a.length) ==> (a(i) != key))
  else
    a(res) == key
})
```

3 Proof construction

Show that list flatMap f flatMap g == list flatMap (x => f(x) flatMap g) using the following axioms:

1. Nil flatMap f == Nil
2. (x :: xs) flatMap f == f(x) ++ (xs flatMap f)
3. Nil ++ xs == xs
4. xs ++ Nil == xs
5. (x :: xs) ++ ys == x :: (xs ++ ys)

Use the proof strategies you have seen in Welder, such as structural induction and equational reasoning. Make sure each step in your reasoning is clearly indicated.

Hint: It may be useful to introduce some auxiliary lemmas.