Functional Programming

Midterm Solution

Friday, November 10 2017

Exercise 1: K-Largest Elements (10 points)

Part I

```
def insert(elem: Int, list: List[Int]): List[Int] = list match {
  case x :: xs if x < elem => x :: insert(elem, xs)
  case _ => elem :: list
}
```

Part II

```
def findKLargestElements(k: Int)(list: List[Int]): List[Int] =
  list.foldLeft(List.empty[Int]) { (acc: List[Int], elem: Int) =>
    val newAcc = insert(elem, acc)

    if (newAcc.size > k) newAcc.tail else newAcc
  }
```

Exercise 2: For comprehensions (10 points)

```
def computeFlips(square: Square): List[Square] = {
  for {
    i <- List(-1, 0, 1)
    j <- List(-1, 0, 1)
    if i != 0 || j != 0
    flip <- computeFlipsInDirection(square.x, square.y, i, j)
  } yield flip
}
```

Exercise 3: Variance (10 points)

T1	?	T2
`A => Y`	`>:`	`A => X`
`A => Y`	`>:`	`B => X`
`FixedChan[A, X]`	`<:`	`FixedChan[A, Y]`
`FixedChan[A, X]`	×	`FixedChan[B, X]`
`FixedChan[A, X]`	`<:`	`Chan[A, Y]`
`Chan[A, Y]`	`>:`	`FixedChan[B, X]`

Exercise 4: Structural Induction (10 points)

We want to show tree.treeMap(f).toList === tree.toList.map(f) for any tree and f of the appropriate types. We proceed by structural induction on tree.

Case `Leaf`:

```
Leaf.toList.map(f)
     === (1) Nil.map(f)
     === (5) Nil
     === (1) Leaf.toList
     === (3) Leaf.treeMap(f).toList
```

Which concludes the case.

Case `Node(left, x, right)`:

Induction hypotheses:

- IH 1 : left.toList.map(f) === left.treeMap(f).toList
- IH 2 : right.toList.map(f) === right.treeMap(f).toList

```
Node(left, x, right).toList.map(f)
     === (2) (left.toList ++ (x ::  right.toList)).map(f)
     === (7) left.toList.map(f) ++ (x ::  right.toList).map(f)
     === (6) left.toList.map(f) ++ (f(x) ::  right.toList.map(f))
     === (IH 1) left.treeMap(f).toList ++ (f(x) ::  right.toList.map(f))
     === (IH 2) left.treeMap(f).toList ++ (f(x) ::  right.treeMap(f).toList)
     === (2) Node(left.treeMap(f), f(x), right.treeMap(f)).toList
     === (4) Node(left, x, right).treeMap(f).toList
```

Which concludes the case and the proof.