Functional Programming

Final Exam Solution

Friday, December 21 2018

Exercise 1: Pure Functional Programming (10 points)

Non tail recursive suolution (7 points)

```
def flatMap[T](list: List[T], f: T => List[T]): List[T] = {
  list match {
    case x :: xs => f(x) ::: flatMap(xs, f)
    case Nil => Nil
  }
}
```

Tail recursive suolution (10 points)

```
def flatMap[T](list: List[T], f: T => List[T]): List[T] = {
  @tailrec def reverse(ls: List[T], acc: List[T]): List[T] = {
    ls match {
      case x :: xs => reverse(xs, x :: acc)
      case Nil => acc
    }
  }
  @tailrec def rec(ls: List[T], acc: List[T]): List[T] = {
    ls match {
      case x :: xs => rec(xs, f(x) ::: acc)
      case Nil => acc
    }
  }
  rec(reverse(list , Nil), Nil)
}
```

Exercise 2: State (10 points)

1. f1(n): **Y**
   all operations used are RT (referentially transparent)
2. f2(n, m): **Y**
   all operations used are RT
3. f3(xs, _ + _): **Y**
   all operations used are RT and the mutable variables are local
4. f3(xs, _ + c.get + _): **N**
   c.get is not RT
5. f4(): **Y**
   all operations used are RT
6. f5(): **N**
   println is not RT
7. f6(c): **N**
   c.inc is not RT
8. f6(new Counter): **N**
   the returned Counter holds state
9. f6(new Counter).get: **Y**
   the Counter is local to the expression (its state does not leak)
10. f7(c): **N**
    c.get is not RT
11. f8(n)(c): **N**
    c.inc is not RT
12. f8(n): **Y**
    the function is not fully applied and the partial application has no state
13. f8(c.get): **N**
    c.get is not RT
14. f9((x:Int) => (), c.get): **Y**
    while c.get is not RT, its result is discarded and does not influence the program
15. f9(f1, f1(c.get)): **N**
    c.get is not RT
16. f9(x => y => println(x+y), 0): **Y**
    the function is not fully applied and the partial application has no state
17. f10(f1): **Y**
    all operations used are RT and the local cache is not observable
18. f10(x => c.inc.get + x): **N**
    c.int and c.get are not RT
19. f10(x => c.get + x): **N**
    c.get is not RT
20. f11: **Y**
    the local counter's state is never changed, so the function passed to f10 in f11 is RT, and f11 is RT

Exercise 3: Lambda Calculus (10 points)

3.1

```
def (succ n)
  (lambda (f x) (f (n f x)))
```

3.2

```
def (size list)
  (list
    zero
    (lambda (h t) (succ (size t)))
  )
```

Exercise 4: Streams (10 points)

```
def trans(src: Stream[Stream[String]], base: Int, n: Int): Stream[String] =
  src.drop(n − 1).head.drop(base − 1).head #:: trans(src, base, n + 1)

def transposed(src: Stream[Stream[String]], x: Int, y: Int): Stream[Stream[String]] =
  trans(src, x, y) #:: transposed(src, x + 1, y)

def transpose(src: Stream[Stream[String]]): Stream[Stream[String]] =
  transposed(src, 1, 1)
```