Федеральное агентство связи

Ордена Трудового Красного Знамени

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

Кафедра Математической Кибернетики и Информационных Технологий

Отчет по лабораторной работе

по предмету «Функциональное программирование»

Выполнил: студент группы

БВТ1802

Самаков Владислав Владимирович

Руководитель:

Мосева Марина Сергеевна

Москва 2020

Задания к работе указаны в файлах с кодом.

Выполнение:

Maps.scala

```scala
/** Напишите вашу реализацию в тестовые функции.
  *
  * https://docs.scala-lang.org/overviews/collections/maps.html
  */
object Maps {

  case class User(name: String, age: Int)

  /* a) В данной Seq[User] сгруппируйте пользователей по имени (`groupBy`) и вычислите средний возраст: `name -> averageAge`
   *    Вы можете реализовать ваше решение в теле тестовой функции. Не изменяйте сигнатуру.
   */
  def testGroupUsers(users: Seq[User]): Map[String, Int] = {
    var groups = users.groupBy(_.name)
    groups.map(x => (x._1, x._2.foldLeft(0)(_ + _.age) / x._2.length))
  }

  /* b) Дана `Map[String, User]` состоящая из имен пользователей `User`, сколько имен пользователей, содержащихся в Map, содержат подстроку "Adam"?
   *    Вы можете реализовать ваше решение в теле тестовой функции. Не изменяйте сигнатуру.
   */
  def testNumberFrodos(map: Map[String, User]): Int = {
    var count = 0
    map.keys.foreach { key =>
      if (map(key).name.contains("Adam")) count += 1
    }
    count
  }

  /* c) Удалите всех пользователей возраст которых менее 35 лет.
   *    Вы можете реализовать ваше решение в теле тестовой функции. Не изменяйте сигнатуру.
   */
  def testUnderaged(map: Map[String, User]): Map[String, User] = {
    map.keys.foreach { key =>
      if (map(key).age < 35) map.-(key)
    }
    map
  }
}
```

Adts.scala

```scala
import scala.util.{Try, Failure, Success}

object Adts {
```

```
  // a) Дан List[Int], верните элемент с индексом n


  // примените функцию из пункта (a) здесь, не изменяйте сигнатуру
  def testGetNth(list: List[Int], n: Int): Option[Int] = Some(list(n))

  // b) Напишите функцию, увеличивающую число в два раза.

  def Double(n: Option[Int]): Option[Int] = if (n.isDefined)  Some(n.get * 2)  else
None

  // примените функцию из пункта (b) здесь, не изменяйте сигнатуру
  def testDouble(n: Option[Int]): Option[Int] = Double(n)

  // c) Напишите функцию, проверяющую является ли число типа Int четным. Если так,
верните Right. В противном случае, верните Left("Нечетное число.").

  def IsEven(n: Int): Either[String, Int] = n % 2 match {
    case 0 => Right(n)
    case 1 => Left("Нечетное число")
  }

  // примените функцию из пункта (c) здесь, не изменяйте сигнатуру
  def testIsEven(n: Int): Either[String, Int] = IsEven(n)

  // d) Напишите функцию, реализующую безопасное деление целых чисел. Верните Right с
результатом или Left("Вы не можете делить на ноль.").

  def SafeDivide(a: Int, b: Int): Either[String, Int] = {
    if (b == 0) Left("Вы не можете делить на ноль")
    else Right(a / b)
  }

  // примените функцию из пункта (d) здесь, не изменяйте сигнатуру
  def testSafeDivide(a: Int, b: Int): Either[String, Int] = SafeDivide(a, b)

  // e) Обработайте исключения функции с побочным эффектом вернув 0.

  def GoodOldJava(impure: String => Int, str: String): Try[Int] = Try(impure(str))

  // примените функцию из пункта (e) здесь, не изменяйте сигнатуру
  def testGoodOldJava(impure: String => Int, str: String): Try[Int] =
GoodOldJava(impure, str)

}
```

Sequence.scala

```
import scala.annotation.tailrec

/** Напишите свои решения в тестовых функциях.
  *
  * Seq(1, 2) match {
  *   case head +: tail => ???
  *   case Nil          => ???
  *   case s            => ???
  * }
  *
  * https://www.scala-lang.org/api/2.12.0/scala/collection/Seq.html
```

```
  */
// Примечание: напишите функции с хвостовой рекурсией

object Sequence {

  /* a) Найдите последний элемент Seq.
   *
   */
  def testLastElement[A](seq: Seq[A]): Option[A] = Some(seq.last)

  /* b) Объедините две Seqs (то есть Seq(1, 2) и Seq(3, 4) образуют Seq((1, 3), (2, 4))) - если Seq длиннее игнорируйте оставшиеся элементы.
   *
   */
  def testZip[A](a: Seq[A], b: Seq[A]): Seq[(A, A)] =  a.zip(b)

  /* c) Проверьте, выполняется ли условие для всех элементов в Seq.
   *
   */
  def testForAll[A](seq: Seq[A])(cond: A => Boolean): Boolean = seq.forall(cond)

  /* d) Проверьте, является ли Seq палиндромом
   *
   */
  def testPalindrom[A](seq: Seq[A]): Boolean = seq.reverse == seq

  /* e) Реализуйте flatMap используя foldLeft.
   *
   */
  def testFlatMap[A, B](seq: Seq[A])(f: A => Seq[B]): Seq[B] = seq.foldLeft(Seq[Seq[B]]())(_ ++ f(_)).flatten
}
```

Strings.scala

```
/** Напишите ваши решения в тестовых функциях.
  *
  * https://www.scala-lang.org/api/2.12.3/scala/collection/immutable/StringOps.html
  */
object Strings {

  /* a) Преобразуйте все символы типа Char в верхний регистр (не используйте заглавные буквы).
   *
   */
  def testUppercase(str: String): String = str.toUpperCase

  /* b) Вставьте следующие значения в строку:
   *       Hi my name is <name> and I am <age> years old.
   *
   */
  def testInterpolations(name: String, age: Int): String = s"Hi, my name is $name and I am $age years old."

  /* c) Добавьте два числа в следующую строку:
   *       Hi,
```

```
   *       now follows a quite hard calculation. We try to add:
   *         a := <value of a>
   *         b := <value of b>
   *
   *         result is <a + b>
   *
   *
   */
  def testComputation(a: Int, b: Int): String = "Hi,\n" +
    "now follows a quite hard calculation. We try to add:\n" +
    s"  a := $a\n" +
    s"  b := $b\n\n" +
    s"  return $a + $b"

  /* d) Если длина строки равна 2, верните всю строку, иначе верните первые два
символа строки.
   */
  def testTakeTwo(str: String): String = str.length match {
    case 2 => str
    case _ => str.substring(0,1)
  }
}
```

Вывод

Я познакомился и научился использовать функционал стандартной библиотеки языка программирования scala, изучил класс String, обработку исключений, контейнеры Seq и Map, а также пару особенностей, не свойственных языку java (Either и Option).