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 № 3

по дисциплине «Функциональное программирование»

на тему:

«Стандартная библиотека Scala»

Выполнил: студент группы БВТ1802

Сурин В.И.

Руководитель:

Мосева Марина Сергеевна

Москва 2020

Выполнение

Код программы

1. Adts

```
import scala.util.{Failure, Success, Try}
/*Реализуйте следующие функции.
  List(1, 2) match {
    case head :: tail => ???
    case Nil => ???
    case l => ???
  }

  Option(1) match {
    case Some(a) => ???
    case None => ???
  }

  Either.cond(true, 1, "right") match {
    case Left(i)  => ???
    case Right(s) => ???
  }

  Try(impureExpression()) match {
    case Success(a) => ???
    case Failure(error) => ???
  }

  Try(impureExpression()).toEither */
object Adts {
  //a) Дан List[Int], верните элемент с индексом n
  def GetNth(list: List[Int], n: Int): Option[Int] = {
    Option(list(n))
  }
  // примените функцию из пункта (a) здесь, не изменяйте сигнатуру
  def testGetNth(list: List[Int], n: Int): Option[Int] = GetNth(list,n)
  //b) Напишите функцию, увеличивающую число в два раза.
  def Double(n: Option[Int]): Option[Int] = {
    n match {
      case Some(a) => Option(a*2)
      case None => n
    }
  }
  // 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] = {
    Either.cond(n % 2 == 0, n, "Нечетное число.") match {
      case Left(i) => Left(i)
      case Right(s) => Right(s)
    }
  }
  // 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] = {
    Try(a/b) match {
      case Success(a) => Right(a)
```

```
      case Failure(error) => Left("Вы не можете делить на ноль.")
    }
  }
  // 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)).toEither match {
      case Left(i) => Success(0)
      case Right(s) => Success(s)
    }
  }
  // примените функцию из пункта (e) здесь, не изменяйте сигнатуру
  def testGoodOldJava(impure: String => Int, str: String): Try[Int] =
GoodOldJava(impure,str)
}
```

2. Maps

```
/**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 map: Map[String, Int]= Map()
    var a=users.groupBy(_.name)
    for (e <- a) {
      var cc= e._2.toBuffer.foldLeft[Int](0)((acc, next) => acc + next.age
      )/e._2.toBuffer.size
      map+=(e._1->cc)
    }
    map
  }
  /*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 = {
    (for (elem <- map if elem._2.name.contains("Adam")) yield elem).size;
  }
  /*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.filter(x => x._2.age >= 35)
  }
}
```

3. Sequence

```
/**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 LastElement[A](seq: Seq[A]): Option[A] = {
    seq match {
```

```
      case last +: Nil  => Option(last)
      case head +: tail => LastElement(tail)
    }
  }
  def testLastElement[A](seq: Seq[A]): Option[A] = LastElement(seq)
  /*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 Zip[A](a: Seq[A], b: Seq[A]): Seq[(A, A)] = {
    def loop[A](a: Seq[A], b: Seq[A], c: Seq[(A, A)]): Seq[(A, A)] = {
      a match {
        case ahead +: atail => b match {
          case blast +: Nil  => c:+(ahead,blast)
          case bhead +: btail => loop(atail,btail,c:+(ahead,bhead))
        }
        case Nil  => c
      }
    }
    loop(a,b,Nil)
  }
  def testZip[A](a: Seq[A], b: Seq[A]): Seq[(A, A)] = Zip(a,b)
  /*c) Проверьте, выполняется ли условие для всех элементов в Seq.*/
  def ForAll[A](seq: Seq[A])(cond: A => Boolean): Boolean = {
    def loop[A](seq: Seq[A],flag: Boolean)(cond: A => Boolean): Boolean = {
      seq match {
        case head :: tail => loop(tail,flag && cond(head))(cond)
        case Nil => flag
      }
    }
    loop(seq,true)(cond)
  }
  def testForAll[A](seq: Seq[A])(cond: A => Boolean): Boolean = ForAll(seq)(cond)
  /*d) Проверьте, является ли Seq палиндромом */
  def Palindrom[A](seq: Seq[A]): Boolean = {
    def loop[A](sseq: Seq[A], aseq: Seq[A]): Boolean = {
      sseq match {
        case head :: tail => loop(tail,aseq = head+:aseq)
        case Nil => seq.equals(aseq)
      }
    }
    loop(seq,Nil)
  }
  def testPalindrom[A](seq: Seq[A]): Boolean = Palindrom(seq)
  /*e) Реализуйте flatMap используя foldLeft.*/
  def FlatMap[A, B](seq: Seq[A])(f: A => Seq[B]): Seq[B] = {
    seq.foldLeft[Seq[B]](Nil)((acc, next) => f(next).++:(acc))
  }
  def testFlatMap[A, B](seq: Seq[A])(f: A => Seq[B]): Seq[B] = FlatMap(seq)(f)
}
```

4. Strings

```
/** 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 = "Hi my name is %s and I
am %d years old.".format(name,age)
```

```
  /*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\t a := %d \n\t b
:= %d \n\n\t result is %d".format(a,b,a+b)
  }
  /*d) Если длина строки равна 2, верните всю строку, иначе верните первые два
символа строки.*/
  def testTakeTwo(str: String): String = str.take(2)
}
```