Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации Ордена Трудового Красного Знамени

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

*Кафедра «Математическая кибернетика и информационные технологии»*

Отчет по лабораторной работе №3

Выполнила: студентка группы БВТ1903

Лаптева Анастасия

Руководитель:

Мосева Марина Сергеевна

**Файл Adts**

// a) Дан List[Int], верните элемент с индексом n  
def GetNth( list:List[Int], n:Int):Option[Int] = list match {  
 case head *::* tail => list.lift(n)//если голова==хвосту  
 case *Nil* => None  
 case list =>list.lift(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*(n) => *Some* (n+n)  
 case None=>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 match {  
 case n => if (n%2==0) *Right*(n)  
 else *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 Div (a:Int, b:Int):Int=a/b\*/  
 def safeDivide (a:Int, b:Int):Either[String, Int] = {  
 /\*Try (Div(a,b)) match {  
 case Success(a) => Right(Div(a,b))  
 case Failure(error) => Left("Вы не можете делить на ноль.")  
 }\*/  
 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)).recover {  
 case \_ => 0  
 }  
  
 // примените функцию из пункта (e) здесь, не изменяйте сигнатуру  
 def testGoodOldJava(impure: String => Int, str: String): Try[Int] = *goodOldJava*(impure, str)

**Результат работы программы:**

Изображение выглядит как текст, снимок экрана, монитор, ноутбук

Автоматически созданное описание

**Файл Maps**

/\* 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] = users.map(User=>(User.name,User.age)).toMap//оьращаем в карту  
def averageAge (users: Map[String, Int]) : Int = users.values.sum/users.size//получаем среднее значение  
  
/\* 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 = map.count(\_.\_2.name.contains("Adam"))  
  
/\* 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(\_.\_2.age>=35)//без age не работает

**Результат работы программы:**

Изображение выглядит как текст, снимок экрана, монитор, внутренний

Автоматически созданное описание

**Файл Sequence**

/\* a) Найдите последний элемент Seq.  
 \*   
 \*/  
@tailrec  
def LastElement[A](seq: Seq[A]): Option[A] = seq match {  
 case last *+: Nil* => *Some*(last) //if 1 element  
 case \_ *+:* tail => *LastElement*(tail) //if many elements  
 case *Nil* => None //if empty  
}  
  
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 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 Palindrome[A](seq: Seq[A]): Boolean = seq.reverse==seq  
def testPalindrome[A](seq: Seq[A]): Boolean =*Palindrome*(seq)  
  
/\* e) Реализуйте flatMap используя foldLeft.  
 \* flatMap преобразует каждый элемент коллекции по оределенному правилу  
 \* foldLeft  
 \*/  
def testFlatMap[A, B](seq: Seq[A])(f: A => Seq[B]): Seq[B] = {  
 def operation(out: Seq[B], in: A): Seq[B] = {  
 var result = f(in)  
 @tailrec  
 def loop(output: Seq[B], input: Seq[B]): Seq[B] = input.length match {  
 case 0 => output  
 case \_ => loop(output :+ input.head, input.tail)  
 }  
 loop(out, result)  
 }  
 seq.foldLeft(*Seq*.empty[B])(operation)  
}

**Результат работы программы:**

Изображение выглядит как текст, снимок экрана, монитор, внутренний

Автоматически созданное описание

**Файл Strings**

/\* a) Преобразуйте все символы типа Char в верхний регистр (не используйте заглавные буквы).  
 \*   
 \*/  
def Uppercase(s:String):String=s.toUpperCase()  
def testUppercase(str: String): String = *Uppercase*(str)  
  
/\* b) Вставьте следующие значения в строку:  
 \* Hi my name is <name> and I am <age> years old.  
 \*   
 \*/  
def Interpolation(name: String, age: Int): String = {  
 return s"Hi my name is **$**name and I am **$**age years old."  
}  
def testInterpolations(name: String, age: Int): String = *Interpolation*(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 Computation(a:Int, b:Int):String={  
 val c = a+b  
 return s"Hi,\n now follows a quite hard calculation. We try to add:\n a := **$**a\n b := **$**b\n result is **$**c"  
}  
def testComputation(a: Int, b: Int): String = *Computation*(a,b)  
  
/\* d) Если длина строки равна 2, верните всю строку, иначе верните первые два символа строки.  
 \*/  
def TakeTwo (s:String):String={  
 val l = s.length  
 if(s==2){return s  
 }else return s.take(2)  
}  
def testTakeTwo(str: String): String = *TakeTwo*(str)

**Результат работы программы:**

Изображение выглядит как текст, снимок экрана, монитор, внутренний

Автоматически созданное описание