**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»

Выполнил: студент группы БВТ1801

Клюшкин Дмитрий Алексеевич

Руководитель:

Мосева Марина Сергеевна

Москва 2020

**Maps**

1. В данной Seq[User] сгруппируйте пользователей по имени (`groupBy`) и вычислите средний возраст: `name -> averageAge`

**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  
}

1. Дана `Map[String, User]` состоящая из имен пользователей `User`, сколько имен пользователей, содержащихся в Map, содержат подстроку "Adam"?

**def** testNumberFrodos(map: Map[String, User]): Int = {  
 (**for** (elem <- map **if** (elem.\_2.name.contains("Adam"))) **yield** elem).size;  
}

1. Удалите всех пользователей возраст которых менее 35 лет.

**def** testUnderaged(map: Map[String, User]): Map[String, User] = {  
 map.filter(x => x.\_2.age>=35)  
}

**Adts**

1. Дан List[Int], верните элемент с индексом n

**def** a(list: List[Int], n: Int): Option[Int] ={  
 *Option*(list(n))  
}

1. Напишите функцию, увеличивающую число в два раза.

**def** b(n: Option[Int]): Option[Int] = {  
 n **match** {  
 **case** *Some*(a) => *Option*(a \* 2)  
 **case** None => n  
 }  
}

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

**def** c(n: Int): Either[String, Int] = {  
 *Either*.*cond*(n%2==0, n, "Нечетное число.") **match** {  
 **case** *Left*(i) => *Left*(i)  
 **case** *Right*(s) => *Right*(s)  
 }  
}

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

**def** d(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.")  
 }  
}

1. Обработайте исключения функции с побочным эффектом вернув 0.

**def** e (impure: String => Int, str: String): Try[Int] = {  
 *Try*(impure(str)).toEither **match** {  
 **case** *Left*(i) => *Success*(0)  
 **case** *Right*(s) => *Success*(s)  
 }  
}

**Sequence**

a) Найдите последний элемент Seq.

**def** funA[A](seq: Seq[A]): Option[A] ={  
 seq **match** {  
 **case** last *+: Nil* => *Option*(last)  
 **case** head *+:* tail => *funA*(tail)  
 }  
}

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** funB[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*)  
}

c) Проверьте, выполняется ли условие для всех элементов в Seq.

**def** funC[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)  
}

d) Проверьте, является ли Seq палиндромом

**def** funD[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*)  
}

e) Реализуйте flatMap используя foldLeft.

**def** funE[A, B](seq: Seq[A])(f: A => Seq[B]): Seq[B]={  
 seq.foldLeft[Seq[B]](*Nil*)((acc, next) => f(next).++:(acc))  
}

**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,\nnow follows a quite hard calculation. We try to add:\n\ta := %d\n\tb := %d\n\n\tresult 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)