Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**
МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/12 Интеллектуальный анализ больших данных в системах поддержки принятия решений**

О Т Ч Е Т
по лабораторной работе № 6

Название: Коллекции

Дисциплина: Языки программирования для работы с большими данными

Студент ИУ6-23М (Группа) _____ (Подпись, дата) М.А. Гейне (И.О. Фамилия)

Преподаватель _____ (Подпись, дата) П.В. Степанов (И.О. Фамилия)

Москва, 2023

ЗАДАНИЕ

Вариант 1:

1. Сложить два многочлена заданной степени, если коэффициенты многочленов хранятся в объекте HashMap.
2. Умножить два многочлена заданной степени, если коэффициенты многочленов хранятся в различных списках.

Вариант 2:

1. Во входном файле хранятся наименования некоторых объектов. Построить список C1, элементы которого содержат наименования и шифры данных объектов, причем элементы списка должны быть упорядочены по возрастанию шифров. Затем “сжать” список C1, удаляя дублирующие наименования объектов.
2. Во входном файле расположены два набора положительных чисел; между наборами стоит отрицательное число. Построить два списка C1 и C2, элементы которых содержат соответственно числа 1-го и 2-го набора таким образом, чтобы внутри одного списка числа были упорядочены по возрастанию. Затем объединить списки C1 и C2 в один упорядоченный список, изменяя только значения полей ссылочного типа.

Задача 1

Для решение задаётся функция, принимающая на вход два HashMap[Int][Int] и возвращающая объект того же типа. Для сложения многочленов требуется сложить коэффициенты совпадающих степеней. Этого возможно добиться функцией `merged`, которая соединит два HashMap, а в случае наложения коэффициентов выполнит pattern-matching; в данном случае в результате матчинга производим сложение коэффициентов. Код решения приведён ниже.

```
import scala.collection.immutable.HashMap

def polySum(a: HashMap[Int, Int], b: HashMap[Int, Int]): HashMap[Int, Int] =
  a.merged(b){ case ((k0, v0), (k1, v1)) => k0 -> (v0 + v1)}

def test() =
  val a = HashMap(0 -> 5, 1 -> 2, 2 -> 3, 4 -> 1)
  val b = HashMap(0 -> 1, 2 -> 2, 3 -> 7)
  println(s"""|Polynomial A:
              |$a
              |Polynomial B:
              |$b
              |A+B:
              |${polySum(a, b)}""".stripMargin)
```

Результат решения задачи представлен ниже.

```
Variant 1
Question 4
Polynomial A:
HashMap(0 -> 5, 1 -> 2, 2 -> 3, 4 -> 1)
Polynomial B:
HashMap(0 -> 1, 2 -> 2, 3 -> 7)
A+B:
HashMap(0 -> 6, 1 -> 2, 2 -> 5, 3 -> 7, 4 -> 1)
------------------------
Developer: mikeGEINE
Task recieved on: Mon Mar 10 19:34:00 MSK 2023
Task completed (this run) on: Fri Apr 28 14:23:28 MSK 2023
```

Задача 2

Для умножения полиномов, коэффициенты которых хранятся в списках, требуется создать пустой массив длины `n+m-1`, затем пройти в циклах оба списка и заполнить массив суммами произведений коэффициентов многочленов. Затем массив приводим к списку. Решение задачи приводится ниже.

```
def multiplyPolynomials(a: List[Int], b: List[Int]): List[Int] =
  val m = a.length
  val n = b.length
  val result = Array.fill(m + n - 1)(0)

  for (i <- 0 until m)
    for (j <- 0 until n)
      result(i + j) += a(i) * b(j)

  result.toList

def test() =
  val a = List(5, 2, 3)
  val b = List(2, 0, -1, 3)

  println(s"""|Polynomial A:
              |$a
              |Polynomial B:
              |$b
              |A*B:
              |${multiplyPolynomials(a, b)}""".stripMargin)
```

Результат работы программы приведён ниже.

```
Variant 1
Question 5
Polynomial A:
List(5, 2, 3)
Polynomial B:
List(2, 0, -1, 3)
A*B:
List(10, 4, 1, 13, 3, 9)
------------------------
Developer: mikeGEINE
Task recieved on: Mon Mar 10 19:34:00 MSK 2023
Task completed (this run) on: Fri Apr 28 14:41:22 MSK 2023
```

Задание 3

Читая файл с именами, программа формирует список пар вида `(имя, хэш имени)`. После этого список сортируется по хэшу, а затем выбираются только уникальные по имени пары. Решение задачи приведено ниже.

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

def readFile(input: String) =
  Using(scala.io.Source.fromFile(input))(_.getLines().toList.map(name => (name,
name.hashCode()))) match
```

```
    case Failure(e) => println(s"Exeption occured when reading input file:
${e.toString()}")
                       Nil
    case Success(value) => value

def test(input: String) =
  val lst = readFile(input)
  val c1 = lst.sortBy((name, code) => code).distinctBy((name, code) => name)

  println(s"""|List C1:
              |$c1""".stripMargin)
```

Результат работы программы приведён ниже.

```
Input:
Scala
Is
The
Is
Best
Language

Output:
Variant 2
Question 4
List C1:
List((Language,-1548945544), (Is,2378), (The,84049), (Best,2066948),
(Scala,79698214))
------------------------
Developer: mikeGEINE
Task recieved on: Mon Mar 10 19:34:00 MSK 2023
Task completed (this run) on: Fri Apr 28 14:47:13 MSK 2023
```

Задача 4

Для решения задачи файл читается и преобразуется к списку `Int`. Список разбивается на первом элементе, не удовлетворяющему условию в функции `span(_ >=0)`. После разбиения два списка сортируются, а после передаются в tail-рекурсивную функцию, которая поэлементно соединяет списки в зависимости от результатов сравнения этих элементов. Когда один из двух исходных списков заканчивается, оставшийся список объединяется с накопленным списком, который при этом разворачивается. Код задачи приведён ниже.

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

```
def readFile(input: String) =
  Using(scala.io.Source.fromFile(input))(_.mkString.split("
").toList.map(_.toInt).span(_ >= 0)) match
    case Failure(e) => println(s"Exeption occured when reading input file:
${e.toString()}");
                       (Nil, Nil)
    case Success(set1, set2) => (set1, set2.tail)

def zipSorted(a: List[Int], b: List[Int]) =
  def recursive(a: List[Int], b: List[Int], acc: List[Int]): List[Int] =
    (a, b) match
      case (ah :: at, bh :: bt) if ah < bh => recursive(at, b, ah :: acc)
      case (ah :: at, bh :: bt) if ah > bh => recursive(a, bt, bh :: acc)
      case (ah :: at, bh :: bt) => recursive(at, bt, ah :: bh :: acc)
      case (Nil, rst) => rst.reverse_:::(acc)
      case (rst, Nil) => rst.reverse_:::(acc)

  recursive(a, b, Nil)


def test(input: String) =
  val sets = readFile(input)
  val c1 = sets._1.sorted
  val c2 = sets._2.sorted
  val zipped = zipSorted(c1, c2)

  println(s"""|C1:
              |$c1
              |C2:
              |$c2
              |Zipped and sorted:
              |$zipped""".stripMargin)
```

Результат работы программы приведён ниже.

```
Input:
4 2 5 7 -11 3 6 5 2

Output:
Variant 2
Question 5
C1:
List(2, 4, 5, 7)
C2:
List(2, 3, 5, 6)
Zipped and sorted:
List(2, 2, 3, 4, 5, 5, 6, 7)
------------------------
Developer: mikeGEINE
Task recieved on: Mon Mar 10 19:34:00 MSK 2023
Task completed (this run) on: Fri Apr 28 14:55:10 MSK 2023
```

ВЫВОДЫ

Изучены способы работы с коллекциями в Scala.

Изучены списки и HashMap.

Изучены способы управления сортировкой списков.

Изучены способы поэлементного построения списков.