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, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ И СИСТЕМЫ УПРАВЛЕНИЯ**

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

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/07 Интеллектуальные системы анализа, обработки и интерпретации больших данных**

О Т Ч Е Т

по лабораторной работе №6

Название: Коллекции

Дисциплина: Языки программирования для работы с большими данными

Студент ИУ6-22М _____ И.Л. Баришпол
(Группа) (Подпись, дата) (И.О. Фамилия)

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

Москва, 2023

Задания

2. Списки (стеки, очереди) I(1..n) и U(1..n) содержат результаты n измерений тока и напряжения на неизвестном сопротивлении R. Найти приближенное число R методом наименьших квадратов.

```
package lab6

import lab6.LeastSquares.findResistance


/**
 * Вариант 1. Задача 2.
 *
 * 2. Списки (стеки, очереди) I(1..n) и U(1..n) содержат
 * результаты n измерений тока и напряжения на неизвестном
сопротивлении R.
 * Найти приближенное число R методом наименьших квадратов.
 */
fun main(args: Array<String>) {
    val I: List<Double> = mutableListOf(1.0, 2.0, 3.0, 4.0,
5.0)
    val U: List<Double> = mutableListOf(2.0, 4.0, 6.0, 8.0,
10.0)
    val R = findResistance(I, U)
    println("Resistance: $R")
}


object LeastSquares {
    fun findResistance(I: List<Double>, U: List<Double>):
Double {
        val n = I.size
        var sumI = 0.0
        var sumU = 0.0
        var sumIU = 0.0
        var sumI2 = 0.0
        for (i in 0 until n) {
```

```
                val iVal = I[i]
                val uVal = U[i]
                sumI += iVal
                sumU += uVal
                sumIU += iVal * uVal
                sumI2 += iVal * iVal
            }
            val denominator = n * sumI2 - sumI * sumI
            if (denominator == 0.0) {
                throw ArithmeticException("Denominator is zero,
cannot compute resistance.")
            }
            val numerator = n * sumIU - sumI * sumU
            return numerator / denominator
        }
    }
```

3. С использованием множества выполнить попарное суммирование произвольного конечного ряда чисел по следующим правилам: на первом этапе суммируются попарно рядом стоящие числа, на втором этапе суммируются результаты первого этапа и т.д. до тех пор, пока не останется одно число.

```
package lab6

import lab6.PairwiseSum.pairwiseSum


/**
 * Вариант 1. Задача 6.
 * 3. С использованием множества выполнить попарное
суммирование произвольного конечного ряда чисел
 * по следующим правилам:
 * на первом этапе суммируются попарно рядом стоящие числа,
 * на втором этапе суммируются результаты первого этапа и
т.д.
 * до тех пор, пока не останется одно число.
 */
fun main() {
```

```
        val numbers: List<Double> = mutableListOf(1.0, 2.0,
3.0, 4.0, 5.0)
        val result = pairwiseSum(numbers)
        println("Pairwise sum: $result")
    }


    object PairwiseSum {
        fun pairwiseSum(numbers: List<Double>): Double {
            var numbers = numbers
            while (numbers.size > 1) {
                val newNumbers: MutableList<Double> =
mutableListOf()
                val iter = numbers.iterator()

                var a = iter.next()
                while (iter.hasNext()) {
                    if (iter.hasNext()) {
                        val b = iter.next()
                        newNumbers.add(a + b)
                        a = b
                    }
                }
                numbers = newNumbers
            }
            return numbers.iterator().next()
        }
    }
```

2. Реализовать класс, моделирующий работу N-местной автостоянки. Машина подъезжает к определенному месту и едет вправо, пока не встретится свободное место. Класс должен поддерживать методы, обслуживающие приезд и отъезд машины.

```
    package lab6

    import java.util.*
```

```
    // 2.     Реализовать класс, моделирующий работу N-местной
автостоянки.
    // Машина подъезжает к определенному месту и едет вправо,
пока не встретится свободное место.
    // Класс должен поддерживать методы, обслуживающие приезд и
отъезд машины.
    fun main() {
        val lot = ParkingLot(10)
        val spot1 = lot.park()
        println("Car parked in spot $spot1")
        val spot2 = lot.park()
        println("Car parked in spot $spot2")
        val spot3 = lot.park()
        println("Car parked in spot $spot3")
        lot.leave(spot2)
        println("Car left from spot $spot2")
        val spot4 = lot.park()
        println("Car parked in spot $spot4")
        lot.leave(spot1)
        println("Car left from spot $spot1")
        lot.leave(spot3)
        println("Car left from spot $spot3")
        lot.leave(spot4)
        println("Car left from spot $spot4")
    }

    class ParkingLot(private val numSpots: Int) {
        private val spots: MutableList<Boolean>

        init {
            spots = ArrayList(Collections.nCopies(numSpots,
false))
        }

        fun park(): Int {
            for (i in 0 until numSpots) {
```

```
                if (!spots[i]) {
                    spots[i] = true
                    return i
                }
            }
            return 0
        }

        fun leave(spot: Int) {
            spots[spot] = false
        }
    }
```

3. Во входном файле хранятся две разреженные матрицы A и B. Построить циклически связанные списки CA и CB, содержащие ненулевые элементы соответственно матриц A и B. Просматривая списки, вычислить: а) сумму S = A + B; б) произведение P = A * B.

```
package lab6

import java.util.*

// 1.    На базе коллекций реализовать структуру хранения чисел с поддержкой следующих операций:
//• добавление/удаление числа;
//• поиск числа, наиболее близкого к заданному (т.е. модуль разницы минимален).
fun main() {
    val storage = NumberStorage()
    storage.add(3)
    storage.add(7)
    storage.add(1)
    storage.add(4)
    storage.add(9)
    println("Numbers in storage: " + storage.getNumbers())
    println("Closest to 6: " + storage.findClosest(6))
    println("Closest to 2: " + storage.findClosest(2))
    println("Closest to 8: " + storage.findClosest(8))
```

```
        storage.remove(4)
        println("Numbers in storage after removing 4: " +
storage.getNumbers())
    }

    class NumberStorage {
        private val numbers: MutableList<Int>

        init {
            numbers = ArrayList()
        }

        fun add(number: Int) {
            numbers.add(number)
        }

        fun remove(number: Int) {
            numbers.remove(Integer.valueOf(number))
        }

        fun findClosest(number: Int): Int {
            check(numbers.isNotEmpty()) { "Number storage is
empty" }
            var closest = numbers[0]
            var minDiff = Math.abs(closest - number)
            for (i in 1 until numbers.size) {
                val diff = Math.abs(numbers[i] - number)
                if (diff < minDiff) {
                    closest = numbers[i]
                    minDiff = diff
                }
            }
            return closest
        }

        fun getNumbers(): List<Int> {
```

```
            return Collections.unmodifiableList(numbers)
        }
    }
```

Вывод: в ходе выполнения данной лабораторной работы были освоены коллекции в Kotlin. Данные задачи помогли закрепить знания по коллекциям и приобрести опыт работы с ними.