БГУИР

Кафедра ЭВМ

Отчет по лабораторной работе № 2

Тема: «Работа со списками и функциями»

Выполнили:

студенты гр. №150502

Альхимович Н.Г.

Скалозуб К.А.

Проверила:

Герман Ю.О.

Минск

2023

**1 ЦЕЛЬ РАБОТЫ**

Изучить технику работы со списками и функциями в Scala.

**2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Примеры функций для работы со списками в Scala.

Map (карта, отображение): эта функция применяет заданную функцию к каждому элементу списка и возвращает новый список с результатами.

object Main22 {

def double(x: Int): Int = x \* 2

def main(args: Array[String]): Unit = {

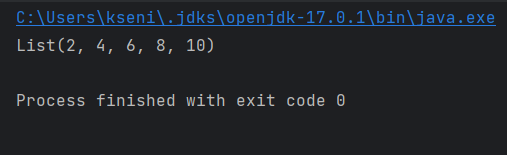
val myList = List(1, 2, 3, 4, 5)

val doubledList = myList.map(double)

println(doubledList) // Output: List(2, 4, 6, 8, 10)

}

}



Поэлементный вывод списка:

object Main22 {

def double(x: Int): Int = x \* 2

def main(args: Array[String]): Unit = {

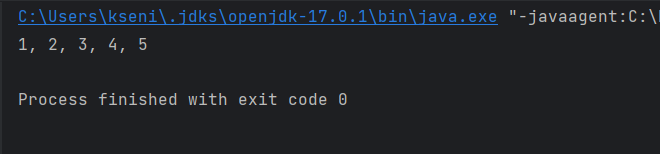
val myList = List(1, 2, 3, 4, 5)

val doubledList = myList.map(double)

println(myList.mkString(", "))

}

}



Filter – функция, которая отбирает элементы списка, удовлетворяющие заданному предикату.

Следующий пример показывает, как вывести четные элементы списка

object Main22 {

def isEven(x: Int): Boolean = x % 2 == 0

def main(args: Array[String]): Unit = {

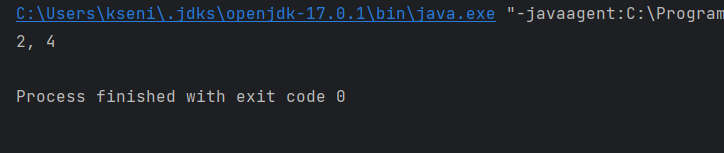
val myList = List(1, 2, 3, 4, 5)

val filteredList = myList.filter(isEven)

println(filteredList.mkString(", ")) // Output: List(2, 4)

}

}



foldLeft – функция, которая последовательно применяется к элементам списка слева направо, накапливая результат. Сумму элементов списка можно найти таким образом:

object Main22 {

def main(args: Array[String]): Unit = {

val myList = List(1, 2, 3, 4, 5)

val sum = myList.foldLeft(0)((ac\_c, x) => ac\_c + x)

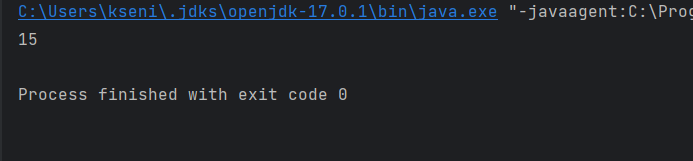
println(sum) // Output: 15

}

}

Здесь переменная ac\_c играет роль аккумулятора. Первоначально ей присваивается значение 0:

myList.foldLeft(0)



zip - функция, объединяющая два списка на примере словаря (dictionary) – ключ-значение.

Возникает вопрос, как взять значение из пары в списке по ключу.

Последовательно покажем решение этой задачи.

object Main22 {

def main(args: Array[String]): Unit = {

val a = List(1, 2, 3)

val b = List("one", "two", "three")

val zipped = a.zip(b) // List((1, "one"), (2, "two"), (3, "three"))

println(zipped)

val filteredList = zipped.filter { case (a, \_) => a == 1 }

println(filteredList)

val tup=filteredList.head

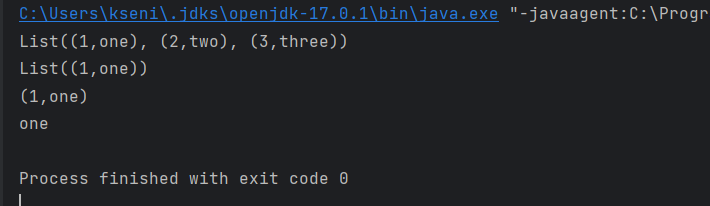
println(tup)

val secondItem = tup.\_2

println(secondItem)

}

}



Обратимся к рекурсивным функциям. Со списками их показывать очень удобно.

Head and tail – функции, возвращающие голову и хвост списка соответственно.

Сумма квадратов элементов списка:

object Main22 {

def sumList(lst: List[Int]): Int = {

if (lst.isEmpty) 0

else lst.head\*lst.head + sumList(lst.tail)

}

def main(args: Array[String]): Unit = {

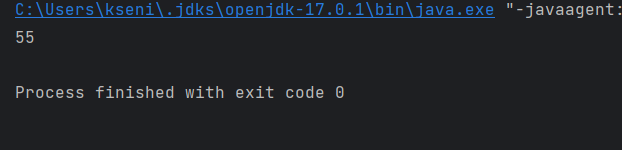
val myList = List(1, 2, 3, 4, 5)

val sumw = sumList(myList)

println(sumw)

}

}



Этот же пример можно реализовать по-другому:

object Main22 {

def main(args: Array[String]): Unit = {

val myList = List(1, 2, 3, 4, 5)

val sumOfSquares = myList.map(item => item \* item).sum

println(sumOfSquares)

}

}

Подсчет числа отрицательных элементов списка (почти предыдущий вариант)

object Main22 {

def main(args: Array[String]): Unit = {

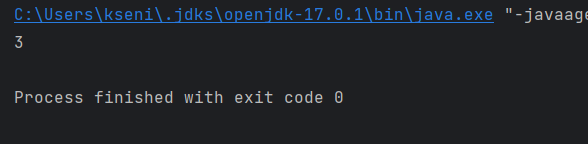
val myList = List(1, -2, 3, -7, -5)

val countNegative = myList.count(item => item < 0)

println(countNegative)

}

}



Отыскание минимального элемента в списке:

object Main22 {

def minList(lst: List[Int], minval: Int): Int = {

if (lst.isEmpty) minval

else if (lst.head<minval) minList(lst.tail,lst.head)

else minList(lst.tail,minval)

}

def main(args: Array[String]): Unit = {

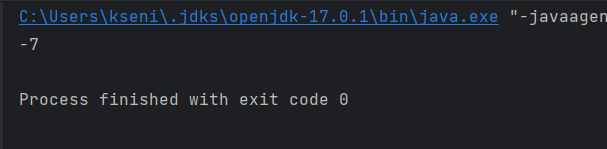
val myList = List(1, -2, 3, -7, -5)

val minw = minList(myList,myList.head)

println(minw)

}

}



Или для ввода списка с клавиатуры:

import scala.io.StdIn

object Main224 {

def minList(lst: List[Int], minval: Int): Int = {

if (lst.isEmpty) minval

else if (lst.head<minval) minList(lst.tail,lst.head)

else minList(lst.tail,minval)

}

def main(args: Array[String]): Unit = {

println("Enter elements of the list (integers), separated by spaces:")

val myList = StdIn.readLine().split(" ").map(\_.toInt).toList

println("Entered list: " + myList)

if (myList.isEmpty) {

println("The list is empty")

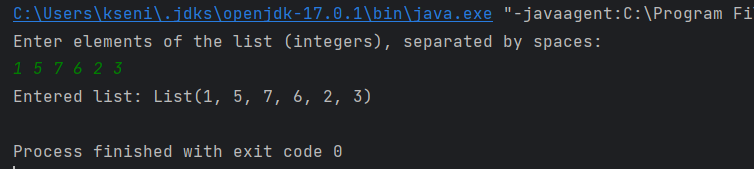
} else

val minw = minList(myList,myList.head)

println(minw)

}

}



Отыскание элемента на позиции *i* при нумерации позиций с нуля, список содержит положительные целые числа. Если индекс выходит за пределы списка, возвращается -1:

object Main22 {

def posList(lst: List[Int], pos: Int): Int = {

if (lst.isEmpty) -1

else if (pos==0) lst.head

else {val newpos=pos-1

posList(lst.tail,newpos)}

}

def main(args: Array[String]): Unit = {

val myList = List(1, 2, 3, 4, 5)

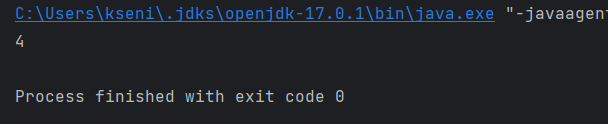
val pos =3

val minw = posList(myList,pos)

println(minw)

}

}



reverse – функция, возвращающая список в обратном порядке

val numbers = List (1, 2, 3, 4, 5)

vale reversed = numbers.reverse // List(5, 4, 3, 2, 1)

**3 ХОД РАБОТЫ**

**Вариант 1**

1. Написать функцию для подсчета суммы элементов списка, значение которых не превосходит 10. Список задать самостоятельно.

Код функции:

def findSumOfChosen(list: List[Int], cond: Int): Int = {

if (list.isEmpty) 0

else {

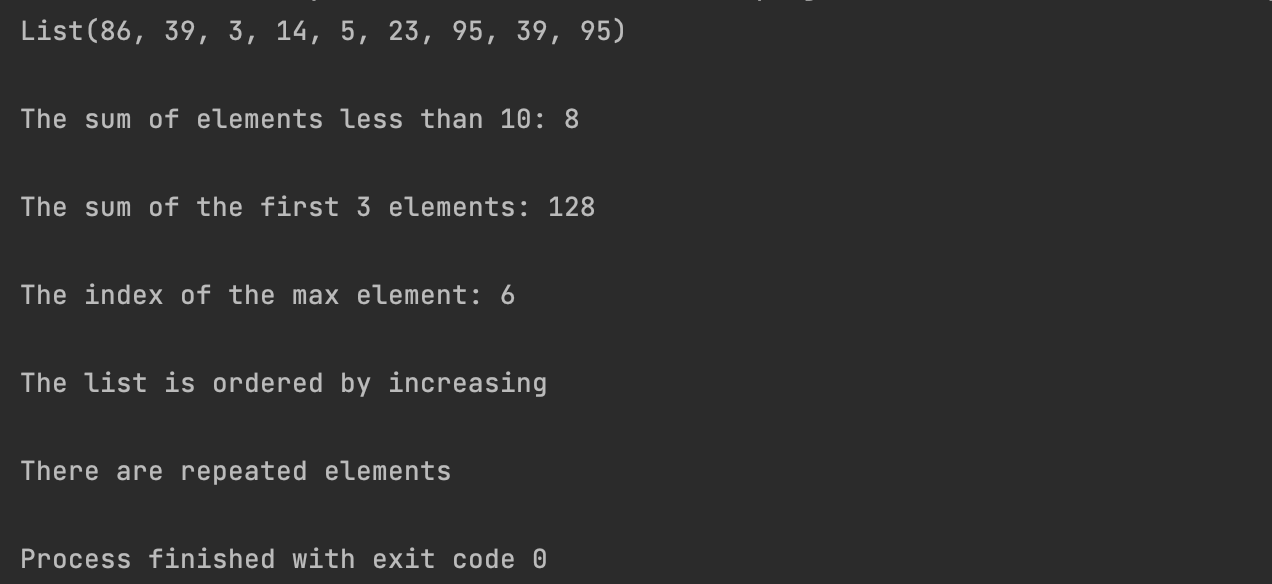
if (list.head < cond) list.head + findSumOfChosen(list.tail, cond)

else findSumOfChosen(list.tail, cond)

}

}

Результат работы программы приведен на рисунке:



1. Написать функцию для подсчета суммы первых трех элементов списка из 10 элементов. Список задать самостоятельно.

Код функции:

def findSumOfFirstThree(list: List[Int], num: Int): Int = {

if (list.isEmpty) 0

else {

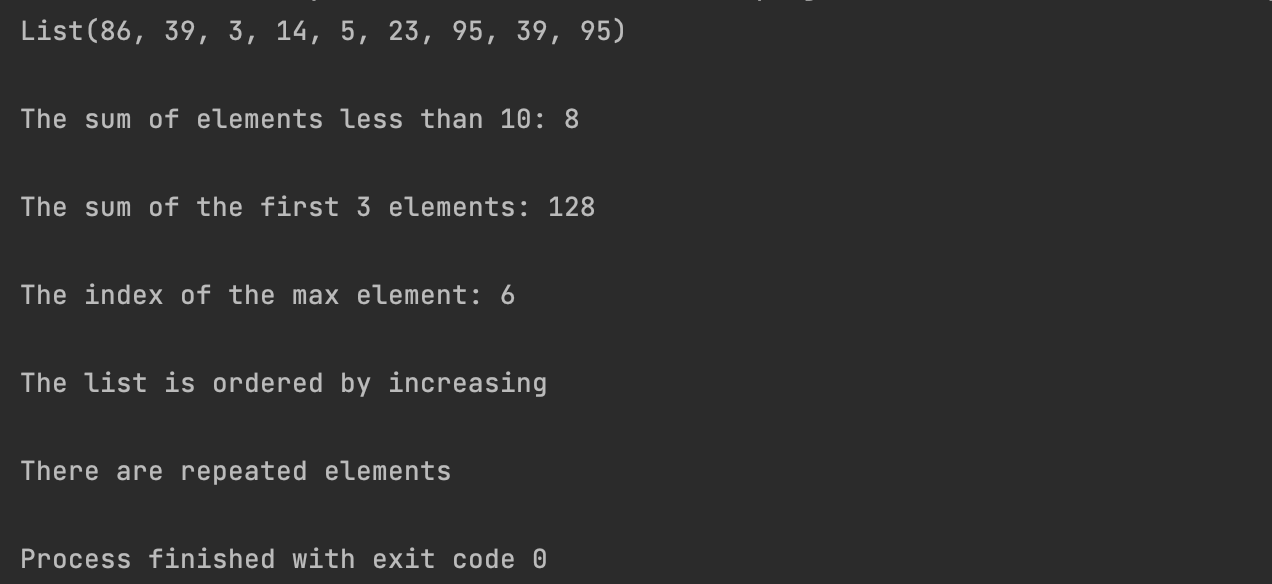
if (list.length > num - 3) list.head + findSumOfFirstThree(list.tail, num)

else 0

}

}

Результат работы программы приведен на рисунке:



1. Написать функцию для отыскания (минимального) индекса максимального элемента списка. Список задать самостоятельно.

Код функции:

def findFirstMax(list: List[Int], max: Int, ind\_current: Int, ind\_max: Int): Int = {

if (list.isEmpty) ind\_max

else {

if (list.head > max) {

findFirstMax(list.tail, list.head, ind\_current + 1, ind\_current)

}

else {

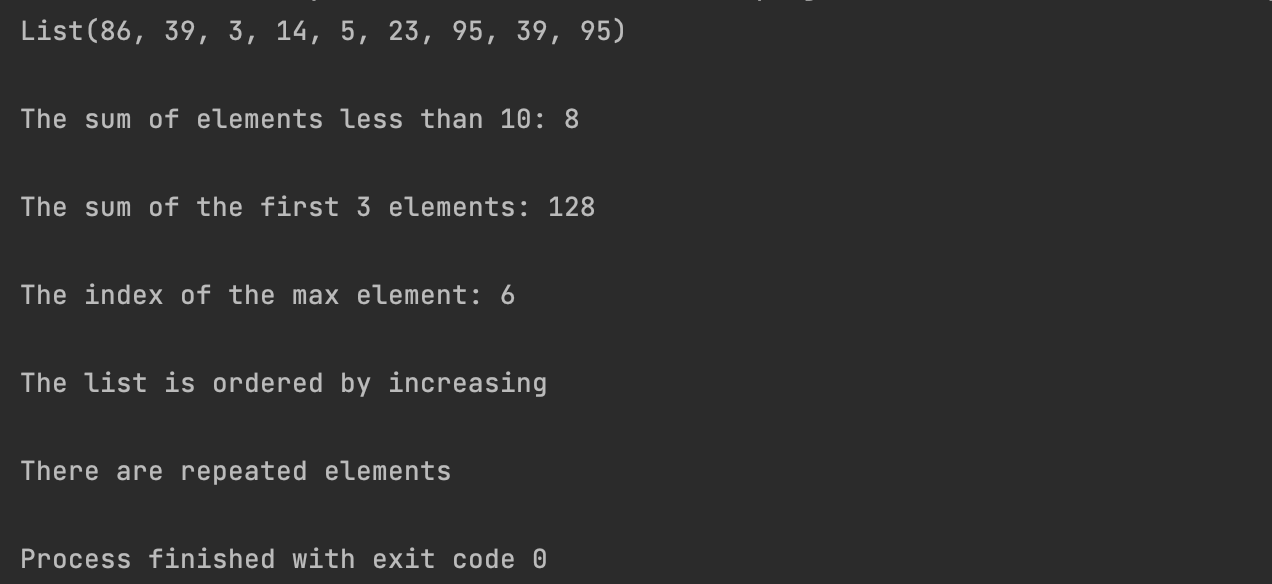
findFirstMax(list.tail, max, ind\_current + 1, ind\_max)

}

}

}

Результат работы программы приведен на рисунке:



1. Написать функцию для проверки того, что список упорядочен по возрастанию. Список задать самостоятельно.

Код функции:

def isOrderedByIncreasing(list: List[Int]): Boolean = {

if (list.isEmpty || list.length == 1) true

else {

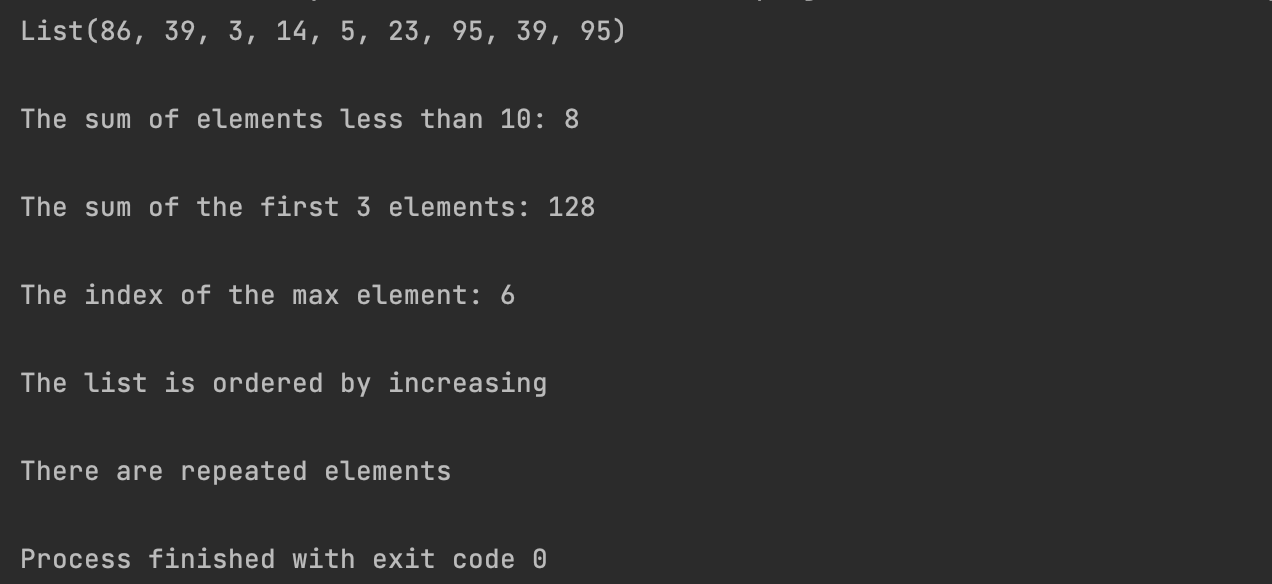
if (list(1) > list.head) isOrderedByIncreasing(list.tail)

else false

}

}

Результат работы программы приведен на рисунке:



1. Написать функцию для проверки наличия одинаковых элементов в списке. Список задать самостоятельно. Функция возвращает значение: да или нет.

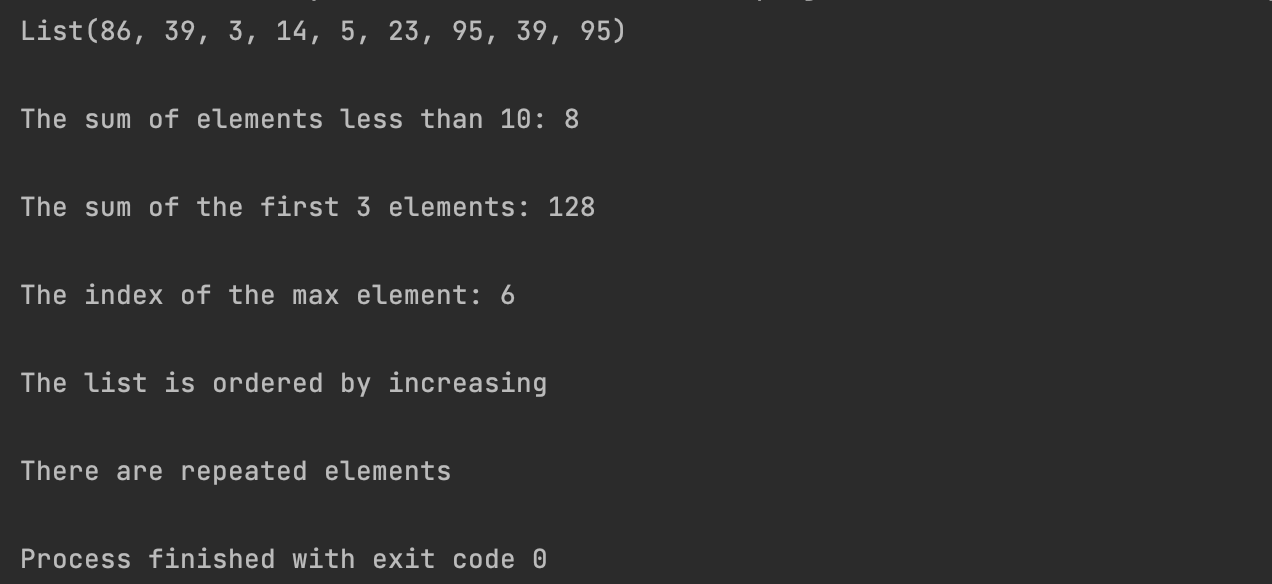
Код функции:

def checkForRepeatedValues(list: List[Int]): Boolean = {

list.groupBy(identity).values.exists(\_.length > 1)

}

Результат работы программы приведен на рисунке:



Код программы:

object Main11 {

def main(args: Array[String]): Unit = {

val list = List(86, 39, 3, 14, 5, 23, 95, 39, 95)

val list2 = List(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)

println(list)

val sum\_particular = findSumOfChosen(list, 10)

println("\nThe sum of elements less than 10: " + sum\_particular)

val sum\_of\_three = findSumOfFirstThree(list, list.length)

println("\nThe sum of the first 3 elements: " + sum\_of\_three)

val ind\_of\_max = findFirstMax(list, list.head, 0, 0)

println("\nThe index of the max element: " + ind\_of\_max)

val is\_ordered = isOrderedByIncreasing(list2)

if (is\_ordered) println("\nThe list is ordered by increasing");

else println("\nThe list isn't ordered")

val repeated\_values = checkForRepeatedValues(list)

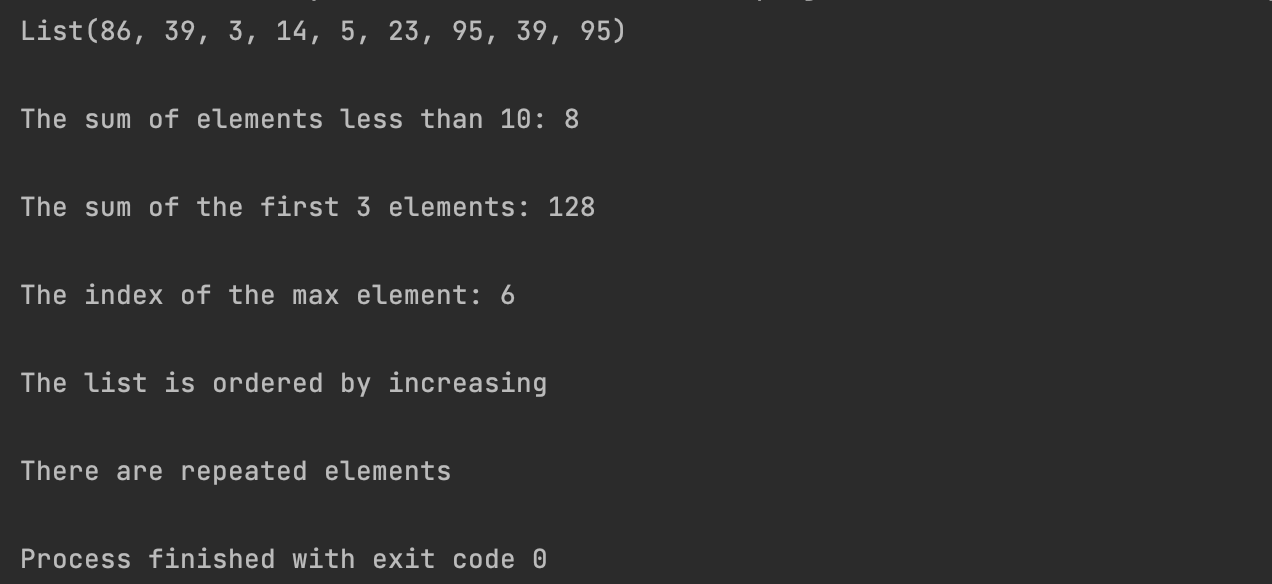
if (repeated\_values) println("\nThere are repeated elements")

else println("\nThere aren't any repeated elements")

}

}

Результат работы программы приведен на рисунке:



**Вариант 2**

1. Написать функцию для подсчета суммы отрицательных элементов списка. Список задать самостоятельно.

Код функции:

def sumOfNegative(list: List[Int]): Int = {

if (list.isEmpty) 0

else {

val current = list.head

if (current<0) {

current + sumOfNegative(list.tail)

} else {

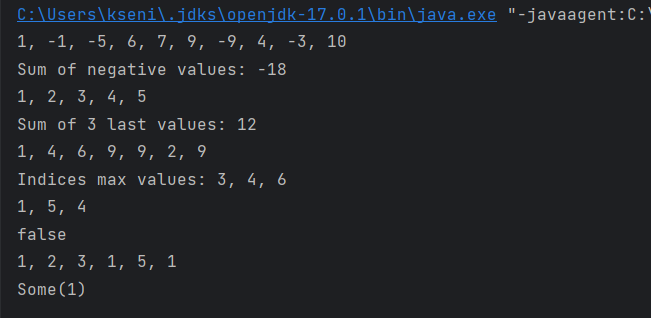
sumOfNegative(list.tail)

}

}

}

Результат работы программы приведен на рисунке:



1. Написать функцию для подсчета суммы последних трех элементов списка из 10 элементов. Список задать самостоятельно.

Код функции:

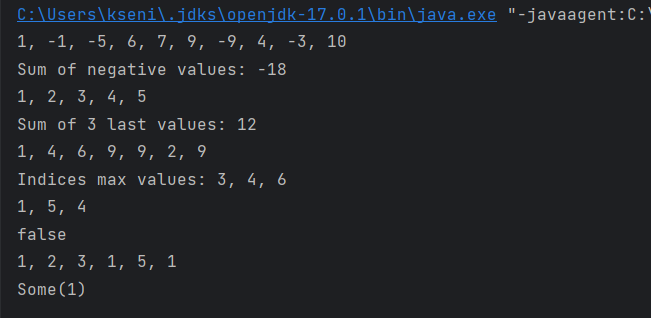
def sumOfThreeLast(list: List[Int]): Int = {

if (list.length<=3) list.sum

else sumOfThreeLast(list.tail)

}

Результат работы программы приведен на рисунке:



1. Написать функцию для отыскания индексов всех максимальных элементов списка. Список задать самостоятельно.

Код функции:

def findAllMaxes(list: List[Int], max: Int, init: List[Int]): List[Int] = {

if (list.isEmpty) {

init.zipWithIndex.collect {

case (element, index) if element == max => index

}

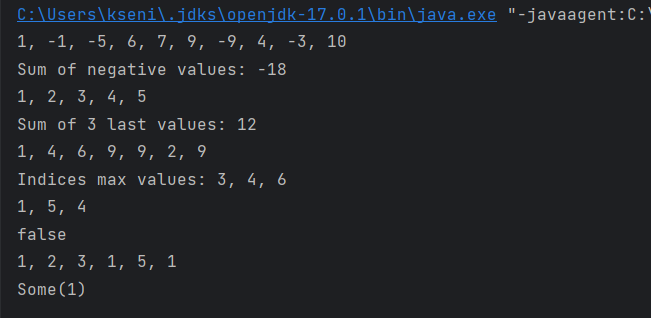
}

else if (list.head > max) findAllMaxes(list.tail, list.head, init)

else findAllMaxes(list.tail, max, init)

}

Результат работы программы приведен на рисунке:



1. Написать функцию для проверки того, что список не упорядочен ни по возрастанию, ни по убыванию. Список задать самостоятельно.

Код функции:

def isUnorderedList(list: List[Int]): Boolean = {

if (list.length < 3) true

else {

val current = list.head

val next = list.tail.head

val rest = list.tail.tail

if ((current <= next) && (next <= rest.head)) {

isUnorderedList(rest)

} else if ((current >= next) && (next >= rest.head)) {

isUnorderedList(rest)

} else {

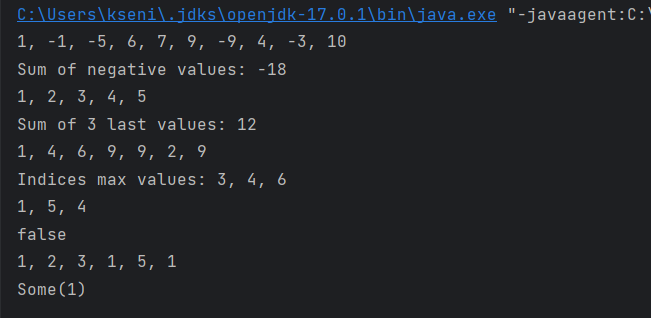
false

}

}

}

Результат работы программы приведен на рисунке:



1. Написать функцию для проверки наличия трех одинаковых элементов в списке. Список задать самостоятельно. Функция возвращает значение такого элемента.

Код функции:

def hasThreeSameElements(list: List[Int]): Option[Int] = {

if (list.isEmpty || list.length<3) None

val head = list.head

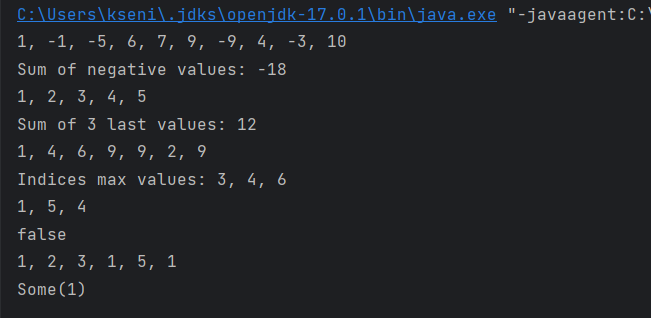
val count = list.tail.count (\_ == head)

if (count == 2) Some(head)

else hasThreeSameElements(list.tail)

}

Результат работы программы приведен на рисунке:



Код программы:

object Main2 {

def main(args:Array[String]):Unit = {

val myList1 = List(1, -1, -5, 6, 7, 9, -9, 4, -3, 10)

println(myList1.mkString(", "))

val sumNegative = sumOfNegative(myList1)

println("Sum of negative values: " + sumNegative)

val myList2 = List(1, 2, 3, 4, 5)

println(myList2.mkString(", "))

val sum3 = sumOfThreeLast(myList2)

println("Sum of 3 last values: " + sum3)

val myList3 = List(1, 4, 6, 9, 9, 2, 9)

maxValuesIndices(myList3)

val myList4 = List(1, 5, 4)

println(myList4.mkString(", "))

val isUnordered = isUnorderedList(myList4)

println(isUnordered)

val myList5 = List(1, 2, 3, 1, 5, 1)

println(myList5.mkString(", "))

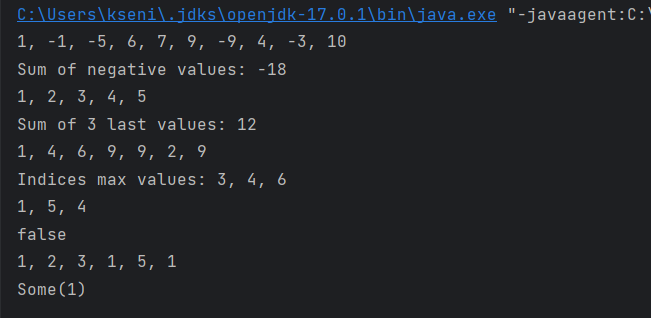
val result = hasThreeSameElements(myList5)

println(result)

}

}

Результат работы программы приведен на рисунке:



**4 ВЫВОДЫ**

В процессе выполнения лабораторной работы была изучена техника работы со списками и функциями в Scala и на основе полученных знаний написан простой S-проект, выполняющий функции над списками.