Министерство цифрового развития, связи и массовых коммуникаций
Ордена Трудового Красного Знамени федеральное государственное
бюджетное
образовательное учреждение высшего образования
«Московский технический университет связи и информатики»

Кафедра Математической кибернетики и информационных технологий

Задачи
по дисциплине «Введение в информационные технологии»

Выполнил студент группы БВТ1901
Мартынов Николай Владимирович
Проверила:
Мосева Марина Сергеевна

Москва
2021

1. Переменные res – это значения val или настоящие переменные var?
val – неизменяемый
var – может быть изменен
res в Scala – val

2. "crazy" * 3 в REPL

```
scala> "crazy"*3
val res0: String = crazycrazycrazy
```

3. Что означает выражение 10 max 2? В каком классе определен метод max?

```
scala> 10 max 2
val res1: Int = 10
```

метод возвращает большее из двух чисел

4. Используя число типа BigInt, вычислите 2^1024

```
scala> BigInt(2).pow(1024)
val res2: scala.math.BigInt =
179769313486231590772930519078902473361797697894230657273430081157732675805500963132708477322407536021120113879871393357658789768814416622492847430639474124377767893424865485276302219601246094119453082952085005768838150682342462881473913110540827237163350510684586298239947245938479716304835356329624224137216
```

5. Что нужно импортировать, чтобы найти случайное простое число вызовом метода probablePrime(100, Random) без использования каких-либо префиксов перед именами probablePrime и Random?

```
import BigInt.probablePrime
import util.Random

scala> probablePrime(100, Random)
val res4: scala.math.BigInt = 93530263821653481395869572398 9
```

6. Один из способов создать файл или каталог со случайным именем состоит в том, чтобы сгенерировать случайное число типа BigInt и преобразовать его в систему счисления по основанию 36, в результате получится строка, такая как "qsnvbevtomcj38o06kul". Отыщите в Scaladoc

методы, которые можно было бы использовать для этого.

```
scala>   probablePrime(100, Random).toString(36)
val res5: String = 31qd41qo0ia5l7q0u43h
```

7. Как получить первый символ строки в языке Scala? А последний символ?

```
scala>  val  s  =  "Martynov"
val s: String = Martynov


scala> s.head
val res6: Char = M


scala> s(0)
val res7: Char = M


scala> s.last
val res8: Char = v


scala> s(s.length - 1)
val res9: Char = v
```

8. Что делают строковые функции take, drop, takeRight и dropRight? Какие преимущества и недостатки они имеют в сравнении с substring?

```
scala> s.take(5)
val res10: String = Marty


scala> s.drop(3)
val res11: String = tynov


scala> s.takeRight(4)
val res12: String = ynov


scala> s.dropRight(4)
val res15: String = Mart
```

9. Сигнум числа равен 1, если число положительное. -1 – если отрицательное, и 0 – если равно нулю. Напишите функцию, вычисляющую это значение.

```
scala> def signum(num:Int){if(num>0)print(1)else if(num<0)print(-1)else print(0)}
                    ^
     warning: procedure syntax is deprecated: instead, add `: Unit =` to explicitly declare
`signum`'s return type
def signum(num: Int): Unit


scala> BigInt(18).signum
val res17: Int = 1


scala> BigInt(-18).signum
val res18: Int = -1


scala> BigInt(0).signum
val res19: Int = 0
```

10. Какое значение возвращает блок {}? Каков его тип?

```
scala> val bl = {}
val bl: Unit = ()
```

11. Напишите на языке Scala цикл, эквивалентный циклу на языке Java
for (int i=10; i>=0; i--) System.out.println(i);

```
scala> for (i <- (0 to 10).reverse) {print(i)}
109876543210
```

12. Напишите процедуру countdown (n: Int), которая выводит числа от n до 0

```
scala> def countdown(n: Int) {
     |     for (i <- n to 0 by -1) println(i)
     | }
```

```
                    ^
      warning: procedure syntax is deprecated: instead, add `: Unit =` to explicitly declare
`countdown`'s return type
def countdown(n: Int): Unit


scala> countdown(18)
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
```

13. Напишите цикл for для вычисления кодовых пунктов Юникода всех букв в строке. Например, произведение символов в строке «Hello» равно 9415087488L.

```
scala> var h:Long = 1
var h: Long = 1


scala> for(i <- "Hello"){ h = h * i.toLong }
```

```
scala> h
val res28: Long = 9415087488
```

14. Решите предыдущее упражнение без применения цикла.
Напишите функцию product(s: String), вычисляющую произведение, как описано в предыдущих упражнениях.

```
scala> def product(s:String):Long={
     |     var h:Long = 1
     |     for(i <- s){
     |         h *= i.toLong
     | }
     |     h
     | }
def product(s: String): Long

scala> product(s)
val res30: Long = 17219016475325280

scala> s
val res31: String = Martynov
```

16. Сделайте функцию из предыдущего упражнения рекурсивной.

```
scala> def productRec(s:String):Long={
     |     if(s.length == 1) return s.charAt(0).toLong
     |     else s.take(1).charAt(0).toLong * product(s.drop(1))
     | }
def productRec(s: String): Long

scala> productRec(s)
val res32: Long = 17219016475325280
```

17. Напишите функцию, вычисляющую xn, где n – целое число.
Используйте следующее рекурсивное определение:
• xn = y2, если n – четное и положительное число, где y=xn/2

• xn = x*xn-1, если n – нечетное и положительное число.
• x0=1.
• xn=1/x-n, если n – отрицательное число.
Не используйте инструкцию return.

```
def xxxx(x:Double,n:Int):Double={
    |    if(n == 0) 1
    |    else if(n>0) x * xxxx(x,n-1)
    |    else 1/xxxx(x,-n)
    | }
def xxxx(x: Double, n: Int): Double


xxxx(5.0,5)
val res34: Double = 3125.0
```

18. f(m,n) - сумма всех натуральных чисел от m до n включительно, в десятичной записи которых нет одинаковых цифр.

```
scala> def sumNatural(n:Int):Int={
    | var sum: Int = (n * (n+1))/2
    | return sum
    | }
    |
def sumNatural(n: Int): Int

scala> def suminRange(l:Int, r:Int):Int={
    |  return sumNatural(r) - sumNatural(l-1)
    | }
def suminRange(l: Int, r: Int): Int


scala> suminRange(2,5)
val res0: Int = 14
```

19. Список содержит целые числа, а также другие списки, такие же как и первоначальный. Получить список, содержащий только целые числа из всех вложенных списков. Пример:
f(List(List(1, 1), 2, List(3, List(5, 8)))) = List(1, 1, 2, 3, 5, 8)

```
scala> def listChisl(ls: List[Any]): List[Any] = ls flatMap {
     | case ms: List[_] => listChisl(ms)
     | case e => List(e)
     | }
def listChisl(ls: List[Any]): List[Any]

scala> listChisl(List(List(1, 1), 2, List(3, List(5, 8))))
val res0: List[Any] = List(1, 1, 2, 3, 5, 8)
```

20. f(n) - сумма цифр наибольшего простого делителя натурального числа n.

```
scala> def largestPrimeFactor(n: Int):Int ={
     | var nN: Int = n
     | var i: Int = 2
     | while(i != nN){
     |     if(nN % i == 0){
     |         nN = nN/i
     |     } else {
     |         i = i+1
     |     }
     | }
     | return nN
     | }
def largestPrimeFactor(n: Int): Int

scala> largestPrimeFactor(15)
val res0: Int = 5

scala> largestPrimeFactor(6)
val res1: Int = 3

scala> largestPrimeFactor(155)
val res2: Int = 31
```

```
scala> def sumOfDig(n: Int):Int ={
     | var nN: Int = largestPrimeFactor(n)
     | var sum: Int = 0
     | while (nN != 0){
     |     sum = sum + nN % 10
     |     nN = nN/10
     | }
     |
     | return sum
     |
     | }
def sumOfDig(n: Int): Int


scala> sumOfDig(155)
val res3: Int = 4
```

21. Список содержит элементы одного, но любого типа. Получить список, содержащий каждый имеющийся элемент старого списка k раз подряд. Число k задается при выполнении программы.

```
scala> def repeatK(k: Int, listIn: List[Any]): List[Any] = {
     | listIn.flatMap(e => List.fill(k)(e))
     | }
def repeatK(k: Int, listIn: List[Any]): List[Any]


scala> repeatK(5, List('M', 'a', 'r', 't', 'y', 'n', 'o', 'v'))
val res0: List[Any] = List(M, M, M, M, M, a, a, a, a, a, r, r, r, r, r, t, t, t, t, t, y, y, y, y, y, n, n, n, n,
n, o, o, o, o, o, v, v, v, v, v)
```

22. f(n) - сумма цифр наибольшего простого делителя натурального числа n.

```
scala> def largestPrimeFactor(n: Int):Int ={
     | var nN: Int = n
     | var i: Int = 2
```

```
     | while(i != nN){
     |     if(nN % i == 0){
     |         nN = nN/i
     |     } else {
     |         i = i+1
     |     }
     | }
     | return nN
     | }
def largestPrimeFactor(n: Int): Int

scala> largestPrimeFactor(15)
val res0: Int = 5

scala> largestPrimeFactor(6)
val res1: Int = 3

scala> largestPrimeFactor(155)
val res2: Int = 31

scala> def sumOfDig(n: Int):Int ={
     | var nN: Int = largestPrimeFactor(n)
     | var sum: Int = 0
     | while (nN != 0){
     |     sum = sum + nN % 10
     |     nN = nN/10
     | }
     |
     | return sum
     |
     | }
def sumOfDig(n: Int): Int

scala> sumOfDig(155)
```

```
val res3: Int = 4
```

23. Список содержит элементы одного, но любого типа. Получить список, содержащий каждый имеющийся элемент старого списка k раз подряд. Число k задается при выполнении программы.

```
scala> def repeatK(k: Int, listIn: List[Any]): List[Any] = {
     | listIn.flatMap(e => List.fill(k)(e))
     | }
def repeatK(k: Int, listIn: List[Any]): List[Any]


scala> repeatK(5, List('M', 'a', 'r', 't', 'y', 'n', 'o', 'v'))
val res0: List[Any] = List(M, M, M, M, M, a, a, a, a, a, r, r, r, r, r, t, t, t, t, t, y, y, y, y, y, n, n, n, n,
n, o, o, o, o, o, v, v, v, v, v)
```

24. f(m,n) - наименьшее общее кратное натуральных чисел m и n.

```
scala> def gcd(a: Int, b: Int):Int ={
     | if (a == 0)
     |     {return b}
     | return gcd(b % a, a)
     | }
def gcd(a: Int, b: Int): Int


scala> def lcm(a:Int, b:Int):Int ={
     | return (a / gcd(a,b))*b
     | }
def lcm(a: Int, b: Int): Int


scala> lcm(15,20)
val res4: Int = 60


scala> lcm(10,20)
val res5: Int = 20
```

25. Список содержит элементы одного, но любого типа. Получить

список, из элементов исходного, удаляя каждый k-й элемент. Число k задается при выполнении программы.

```
scala> def deleteK(k: Int, listIn: List[Any]): List[Any] = {
     |     listIn.zipWithIndex.filter(pair => (1 + pair._2) % k != 0).map(_._1)
     | }
def deleteK(k: Int, listIn: List[Any]): List[Any]


scala> deleteK(5, List('M', 'a', 'r', 't', 'y', 'n', 'o', 'v'))
val res1: List[Any] = List(M, a, r, t, n, o, v)
```

26. f(n,k) - число размещений из n по k. Факториал не использовать.

```
scala> def comb(n: Int, k: Int):Int ={
     |  if((n==k) || (k==0)) {return 1}
     |  if(k==1) {return n}
     |  return comb(n-1,k)+comb(n-1,k-1)
     | }
def comb(n: Int, k: Int): Int


scala> comb(4,2)
val res0: Int = 6


scala> comb(7,5)
val res1: Int = 21
```

27. Список содержит элементы одного, но любого типа. Получить новый список, перемещая циклически каждый элемент на k позиций влево (при перемещении на одну позицию первый элемент становится последним, второй первым и так далее). Число k задается при выполнении программы. Если k отрицательное, то перемещение происходит вправо.

```
scala> def moveL[Any](k: Int, listIn: List[Any]): List[Any] = {
     | val kK = if (listIn.isEmpty) 0 else k % listIn.length
     | if (kK < 0) moveL(kK + listIn.length, listIn)
     | else (listIn drop kK) ::: (listIn take kK)
     | }
```

```
def moveL[Any](k: Int, listIn: List[Any]): List[Any]


scala> moveL(4, List('M', 'a', 'r', 't', 'y', 'n', 'o', 'v'))
val res2: List[Char] = List(y, n, o, v, M, a, r, t)


scala> moveL(-3, List('M', 'a', 'r', 't', 'y', 'n', 'o', 'v'))
val res3: List[Char] = List(n, o, v, M, a, r, t, y)
```

28. f(n) - наибольшее совершенное число не превосходящее n. Совершенным называется натуральное число n равное сумме своих делителей, меньших n, например 6 = 1 + 2 + 3 (f(6) = 6, f(7) = 6, ...).

```
scala> def maxPerf(n:Int):Unit={
     | for(i<-(1 to n)){
     | var sum = 0;
     |    for(k <- (1 to i))
     |    {if(i%k==0 && i!=k)
     |        {sum=sum+k;}
     |    };
     |    if(sum==i) {println(i)}
     | }
     | }
def maxPerf(n: Int): Unit


scala> maxPerf(6)
6


scala> maxPerf(7)
6


scala> maxPerf(30)
6
28
```

29. Список содержит элементы одного, но любого типа. Получить два списка из элементов исходного, выбирая в первый элементы с четными

индексами, а во второй с нечетными.

```
scala> import scala.collection.mutable.ListBuffer
import scala.collection.mutable.ListBuffer

scala> def indexL(listIn: List[Any]):   List[List[Any]] = {
     | var a = new ListBuffer[Any]
     | var b = new ListBuffer[Any]
     | (0 to listIn.size - 1).foreach(x => {
     |     if (x % 2 == 0) { a += listIn(x); }
     |     else { b += listIn(x); }
     | })
     | List(b.toList, a.toList)
     | }
def indexL(listIn: List[Any]): List[List[Any]]

scala> indexL(List(1,2,3,4))
val res0: List[List[Any]] = List(List(2, 4), List(1, 3))

scala> indexL(List('M', 'a', 'r', 't', 'y', 'n', 'o', 'v'))
val res1: List[List[Any]] = List(List(a, t, n, v), List(M, r, y, o))
```

30. f(n) - наибольшее из чисел от 1 до n включительно, обладающее свойством: сумма цифр n в некоторой степени > 1 равна самому числу n. Пример: 512 = 83

```
scala> def maxN(n: Int): Int ={
     | (1 to n).filter(i =>{
     |     var x = i.toString.foldLeft(0: Int)((x,y) => x + y - 48)
     |     x = x * x
     |     while (x < i && x > 1) x = x * x
     |     x == i
     | }).max
     | }
def maxN(n: Int): Int

scala> maxN(1)
```

```
val res0: Int = 1


scala> maxN(511)
val res1: Int = 81
```

31. Список в качестве элементов содержит кортежи типа: (n, s), где n — целые числа, а s — строки. Получить два списка из элементов исходного, выбирая в первый числа, а во второй строки из кортежей.

```
scala> import scala.collection.mutable.ListBuffer
import scala.collection.mutable.ListBuffer


scala> def twoL(listIn: List[Tuple2[Int, String]]):   List[List[Any]] = {
     | var a = new ListBuffer[Int]
     | var b = new ListBuffer[String]
     | (0 to listIn.size - 1).foreach(x => {
     |     a += listIn(x)._1;
     |     b += listIn(x)._2
     | })
     | List(a.toList, b.toList)
     | }
def twoL(listIn: List[(Int, String)]): List[List[Any]]


scala> twoL(List((4, "aa"), (3, "b"), (2, "c"), (1, "fff"), (7, "m"), (18, "Martynov")))
val res0: List[List[Any]] = List(List(4, 3, 2, 1, 7, 18), List(aa, b, c, fff, m, Martynov))
```