Лабораторна робота 1: Функції першого порядку, рекурсія і списки на мові Scala

Функції першого порядку + рекурсія

[3 бали] Визначити функції not, and, or, nand, xor, nor, які повертають логічне значення, наприклад:

```
scala> not(false)
res0: Boolean = true

scala> and(false, true)
res1: Boolean = false

scala> or(false, true)
res2: Boolean = true

scala> xor (false, true)
res3: Boolean = true

scala> nor (false, true)
res4: Boolean = false

scala> nand (true, true)
res5: Boolean = false
```

Напишіть функцію, яка виводить таблицю істинності для певної логічної функції
`table (f: (Boolean, Boolean) =>  Boolean)`

```
scala> table (and)
A       B       result
true    true    true
true    false   false
false   true    false
false   false   false
```

[2 бали] Визначить, чи є два додатних числа взаємнопростими значеннями

```
scala> isCoprime (35, 64)
res0: Boolean = false
```

[1 бал] Визначить, чи є число простим

```
scala> isPrime (7)
res0: Boolean = true
```

[1 бал] Побудуйте список з простих чисел, розташованих в певному діапазоні

```
scala> listPrimesInRange (7 to 31)
res0: List [Int] = List (7, 11, 13, 17, 19, 23, 29, 31)
```

[3 бали] У трикутнику Паскаля знайти число за номером стовпчика і рядка, де знаходиться це число

```
def pascal(c: Int, r: Int): Int

>scala pascal(0,2)
res0: Int = 1

>scala pascal(1,2)
res0: Int = 2

>scala pascal(1,3)
res0: Int = 3
```

[3 бали] Є вираз, який містить дужки. Визначити, чи є він збалансованим, тобто порядок дужок, що відкриваються відповідає дужкам, що закриваються

```
def balance(chars: List[Char]): Boolean

>scala balance("(if (zero? x) max (/ 1 x))")
       res0: Boolean = true

>scala balance(":-)")
       res0: Boolean = false
```

Списки

[1 бал] Знайти останній елемент списку
Приклад роботи:

```
scala> last(List(1, 1, 2, 3, 5, 8))
res0: Int = 8
```

Надати дві реалізації:

1) застосування функції Scala
2) написати свою рекурсивну функцію з хвостовою рекурсією

[1 бал] Знайти елемент списку в позиції n. Надати дві реалізації
Приклад роботи:

```
scala> nth(2, List(1, 1, 2, 3, 5, 8))
res0: Int = 2
```

Списки + рекурсивні функції

[2 бали] Побудуйте список (упорядкований за спаданням) з простих множників додатнього числа

```
scala> primeFactors (315)
res0: List[Int] = List(3, 3, 5, 7)
```

[2 бали] Побудуйте список (упорядкований за спаданням) з простих множників додатнього числа та їх кількості

```
scala> primeFactors (315)
res0: List[(Int, Int)] = List((3, 2), (5,1), (7,1))
```

[1 бали] Приберіть зі списку всі однакові елементи, які розташовані поряд

```
scala> compress (List (‘a’, ‘a’, ‘a’, ‘b’, ‘c’, ‘c’, ‘a’, ‘a’, ‘d’, ‘e’, ‘e’,
‘e’, ‘e’)
res0: List[Symbol] = List(‘a’, ‘b’, ‘c’, ‘a’, ‘d’, ‘e’)
```