SV-44 KOT57

Sa se scrie un program Kotlin care avand la intrare doua multimi A si B de cate 15 numere initilizate aleator care vor fi depuse in niste colectii sa se aplice utilizand calculul lambda si operatorii specifici urmatoarea transformare AxB={(a,b)|a apartine lui A si b apartine lui B} iar rezultatul trebuie la randul lui depus intr-o colectie si aceasta afisata.

```
fun main(args:Array<String>) {

    val A = mutableListOf<Int>()
    val B = mutableListOf<Int>()

    for (i in 1..15) {
        A.add((0..100).random())
        B.add((0..100).random())
    }

    println(A)
    println(B)

    var AxB = A.map { x->B.map { y->Pair(x,y) } }
    println(AxB)
}
```

SV46-Kot59

Pornind de la doua multimi A si B care contin 20 de elemente depuse in doua colectii separate si tinand cont de AxB={(a,b)|a apartine lui A si b apartine lui B}, sa se scrie un program Kotlin care va calcula (AXB) intersectat cu (BXA) utilizand functii specifice colectiilor si eventual lambda calcul urmatoarele. Rezultatul este depus intr-un dictionar iar acesta va fi afisat.

```
fun main(args: Array<String>) {
    val A = listOf(1, 2, 3)
    val B = listOf(2, 3, 4)

    println(A)
    println(B)

    val AxB = A.map { a -> B.map { b -> Pair(a, b) } }.flatMap { it }.distinct()
    val BxA = B.map { b -> A.map { a -> Pair(b, a) } }.flatMap { it }.distinct()

    println(AxB)
    println(BxA)

    val intersect = AxB.filter{
        BxA.contains(it)
    }
    println(intersect)

    val myMap = intersect.groupBy { it.first }.map {
```

```
        Pair(it.key, it.value.map {index ->
            index.second
        })
    }.toMap()
```

SV47-KOT60

Pornind de la doua multimi A si b care contin 20 de elemente depuse in doua colectii separate si tinand cont de A X B={(a,b)|a apartine lui A si b apartine lui B}, sa se scrie un program Kotlin care va calcula (A X B) reunit cu (B X B) utilizand functii specifice colectiilor si eventual lambda calcul urmatoarele. Rezultatul este depus intr-un dictionar iar acesta va fi afisat.

```
fun main(args:Array<String>) {

    val A =ListOf(1,2,3)
    val B = ListOf(2,3,4)
    println(A)
    println(B)

Val AxB=A.map { a->B.map { b-> Pair(a, b) } }.flatMap { it }.distinct()

Val BxB=B.map { b->B.map { b-> Pair(b,b) } }.flatMap { it }.distinct()

Val reunion= AxB.filter { !BxB.contains(it) }
reunion += (BXB.filter { BXB.contains(it) })

Println(reunion)
```

```
val myMap = reunion.groupBy { it.first }.map {
        Pair(it.key, it.value.map {index ->
            index.second
        })
    }.toMap()
```

Sv50-Kot62

Pornind de la doua multimi A si B care contin 20 de elemente depuse in doua colectii separate si tinand cont de AxB={(a,b)|a apartine lui A si b apartine lui B}, sa se scrie un program Kotlin care va calcula (A reunit cu B) x (B intersectat cu A) utilizand functii specifice colectiilor si eventual lambda calcul urmatoarele. Rezultatul este depus intr-un dictionar iar acesta va fi afisat.

```
val A =ListOf(1,2,3)
 val B = ListOf(2,3,4)
println(A)
println(B)

Val AuB= A.filter { !B.contains(it) }
AuB += B.filter { B.contains(it) }


val Intersect = B.filter{ A.contains(it) }
```

```
Val AuBxIntersect= AuB.map { a->Intersect.map { b->Pair(a,b) } }.flatMap {it
}.distinct


val myMap = AuBxIntersect.groupBy { it.first }.map {
      Pair(it.key, it.value.map {index ->
         index.second
      })
   }.toMap()
```

SV53-KOT65

Fie doua colectii initializate cu 100 de numere din submultimile A={x apartine lui N| x=(8n-18)/2n-9, n apartine lui N} si B={x apartine lui Z|x=(9n^2-48n+16)/(3n-8), n apartine lui N}. Pentru acestea se vor calcula utilizand functii de transformare specifice si lambda calcul urmatoarele operatii (AxB)U(B intersectat cu A) unde AxB={(a,b)|a apartine lui a si b apartine lui B}. Rezultatul este depus intr-un hasmap iar acesta va fi afisat.

SV32-KOT38

Utilizand Kotlin sa se aplice un functor peste elementele unui hasmap care va realizare urmatoarele procesari va aplica f(x)=3x-1 ca lambda le va transforma in string si va afisa rezultatul.

```
fun main(args:Array<String>) {
    var my_map = hashMapOf<Int, Int>()
    for (i  in 0..100)
    {
        my_map[i] = i
    }
    println(my_map)
    my_map.values.map { i-> 3 * i - 1  }
        .map(Int::toString)
        .forEach ( ::println )
}
```

SV33-KOT39

Utilizand Kotlin functiile pentru procesare submultimi din care se va prelua un fisier text(creat de programator cu cateva propozitii in el) si se vor extrage doua caractere din mijlocul cuvantului daca cuvantul are minim patru caractere. Se va utiliza combinatie cu lambda peste colectii pentru procesare.

```
import java.io.File

fun main(args: Array<String>) {
    val myFile = File("file.in")

    val myText = myFile.readText()
```

```
    println(myText)

    val result = myText.split(" ").filter{
        it.length <= 4
    }.map {
        val size = it.length/2-1
        if(it.length % 2 == 1)
            it[size+1]
        else {
            (it[size].toString() + it[size + 1].toString())
        }
    }

    println(result)
}
```

SV39-KOT53

Fie o multime la intrare A de 100 de elemente pare alese aleator care sunt depuse intr-o colectie. Sa se scrie un program in care va calcula Bn=a1^2+a2^2+....+an^2, oricare ar fi n apartinand lui N* pentru orice valoare n in respectivul interval.

```
import java.lang.Exception
import java.util.*
import kotlin.random.Random

fun main() {
```

```
    val nrValues = 10

    val list = List(10){Random.nextInt(0, 10) * 2}

    val buffer = Scanner(System.`in`)
    print("Inceput de interval: ")
    val start = buffer.nextInt()
    print("Final de interval: ")
    val end = buffer.nextInt()

    if(start > end)
        throw Exception("Valoarea de inceput trebuie sa fie mai mica decat cea de final")

    if(end >= nrValues || start < 0)
        throw Exception("Capetele intervalului trebuie sa fie numere pozitive mai mici strict decat 100")

    val b = ArrayList<Int>()

    // calcul primul i
    var myValue = 0
    for(i in 0 until start)
        myValue += list[i]*list[i]

    b.add(myValue)

    for(i in start until end) {
        myValue += list[i]*list[i]
```

```
        b.add(myValue)
    }

    println("list is $list")

    b.forEachIndexed{index, value ->
        println("${index + start} calculated is $value")
    }

}
```

SV76-KOT106

Utilizand Kotlin si OOP sa se creeze un program care permite modelarea unei sali de laborator cu tot cu operatii pentru obiecte. Se vor folosi liste mutable. Se va desena diagrama UML.

```
import java.lang.Exception
import kotlin.reflect.jvm.internal.impl.types.AbstractTypeCheckerContext

open class Om(val nume:String){
    fun Scrie_Tabla(){
        println(nume + " scrie tabla")
    }
    fun Sterge_Tabla()
    {
```

```
        println(nume + " sterge tabla")
    }

}

class Elev( nume:String) : Om(nume)
{
    fun Asculta_De_Profesor()
    {
        println("Elevul " + nume + " asculta")
    }
}

class Profesor( nume:String) : Om(nume)
{
    fun Preda_La_elev()
    {
        println("Profesorul " + nume + " preda")
    }
    fun Porneste_Calculatoare( calculatoare: MutableList<Calculator>)
    {
        calculatoare.forEach { it.Start() }
    }
}

class Calculator(val id: Int)
{
    fun Start()
```

```
    {
     println("S-a pornit calculatorul: "  + id)
    }
}

class Sala(val nume: String, val elevi: MutableList<Elev>, val p:Profesor, val calculatoare: MutableList<Calculator>)
{
        fun Actioneaza(om:String, actiune: String)
        {
            when(om)
            {
                "elev"-> {

                    when(actiune)
                    {
                        "asculta"-> elevi.forEach { it.Asculta_De_Profesor() }
                        "sterge" -> elevi.forEach{it.Sterge_Tabla()}
                        "scrie" -> elevi.forEach{it.Scrie_Tabla()}

                    }

                }

                "profesor"-> {

                    when(actiune)
                    {
```

```
                    "preda"->p.Preda_La_elev()
                    "sterge" -> p.Sterge_Tabla()
                    "scrie" -> p.Scrie_Tabla()
                    "start" -> p.Porneste_Calculatoare(calculatoare)
                }

            }
        }
    }
}
fun main(args:Array<String>) {
    val alex= Elev("Alex")
    val marian= Elev("Marian")
    val hara= Elev("Hara")
    val m= mutableListOf<Elev>()
    m.add(alex)
    m.add(hara)
    m.add(marian)
    val c1=Calculator(1)
    val c2=Calculator(2)
    val c3=Calculator(3)
    val calculatoare = mutableListOf<Calculator>()
    calculatoare.add(c1)
    calculatoare.add(c2)
    calculatoare.add(c3)
    val p=Profesor("George")
    val sala=Sala("Sala2",m,p,calculatoare)
```

```
    sala.Actioneaza("elev","asculta")
    sala.Actioneaza("profesor","sterge")

}
```