**Amrita School of Engineering**

**Department of Computer Science and Engineering**

**19CSE313 – Principles of programming Languages**

**Practice Assignment**

**Date: 25/04/2022** **Topic: Scala**

1. **Write a functional Scala program nestedRember to remove a given element in a nested list. The function should remove all occurrences of the element even if it is inside a sub-list.**

**Code:**

object practice {

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

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

    println(nestedRember(list, 3))

  }

  def nestedRember(*list*: List[Any], *element*: Any): List[Any] = {

    list match {

      case Nil => Nil

      case head :: tail =>

        head match {

          case *subList*: List[Any] => nestedRember(subList, element) :: nestedRember(tail, element)

          case \_ => if (head == element) nestedRember(tail, element) else head :: nestedRember(tail, element)

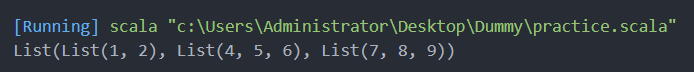
        }

    }

  }

}

**Output:**

****

1. **The interleave function interleaves two given lists like the following.**

**interleave((1,2,3), (-1,-2,-3)) returns (1,-1,2,-2,3,-3). Write a functional Scala program nestedInterleave that accepts nested lists and returns interleaved sub-lists.**

**Code:**

def nestedInterleave(*l1*: List[Any], *l2*: List[Any]): List[Any] = {

  (l1, l2) match {

    case (Nil, Nil) => Nil

    case (Nil, \_) => l2

    case (\_, Nil) => l1

    case (h1::t1, h2::t2) => h1::h2::nestedInterleave(t1, t2)

  }

}

1. **NestedSetUnion:**

**Code:**

def nestedSetUnion(*list1*: List[Any], *list2*: List[Any]): List[Any] = {

  (list1, list2) match {

    case (Nil, \_) => list2

    case (\_, Nil) => list1

    case (x :: xs, y :: ys) =>

      if (x == y) nestedSetUnion(xs, ys)

      else x :: nestedSetUnion(xs, ys)

  }

}

println(nestedSetUnion(((1,2),3,4), ((1,2),(3,4))))