Saniha C N
1NT22CS179

1. Write the scala code to implement bubble sort algorithm.
CODE :

```
object BubbleSort {
  def bubbleSort(arr: Array[Int]): Unit = {
    val n = arr.length

    for (i <- 0 until n) {

      for (j <- 0 until n - i - 1) {

        if (arr(j) > arr(j + 1)) {
          val temp = arr(j)
          arr(j) = arr(j + 1)
          arr(j + 1) = temp
        }
      }
    }
  }

  def main(args: Array[String]): Unit = {
    val arr = Array(64, 34, 25, 12, 22, 11, 90)
    println("Original array:")
    println(arr.mkString(" "))

    bubbleSort(arr)

    println("Sorted array:")
    println(arr.mkString(" "))
  }
}
```

Output:

```
Original array:
64 34 25 12 22 11 90
Sorted array:
11 12 22 25 34 64 90
```

2. Write scala code to find the length of each word from the array.

CODE :

```
object WordLengths {
  def main(args: Array[String]): Unit = {
    val words = Array("Scala", "is", "powerful", "and", "concise")
    val lengths = words.map(_.length)
    for ((word, length) <- words zip lengths) {
      println(s"'$word' has length $length")
    }
  }
}
```

Output:

```
'Scala' has length 5
'is' has length 2
'powerful' has length 8
'and' has length 3
'concise' has length 7
```

3. Write scala code to find the number of books published by each author, referring to the collection given below, with (authorName, BookName)

{ (‘ Dr. Seuss’: ‘How the Grinch Stole Christmas!’) ,(‘ Jon Stone’: ‘Monsters at the End of This Book’) , (‘ Dr. Seuss’: ‘The Lorax’) ,(‘Jon Stone’: ‘Big Bird in China’) (‘ Dr. Seuss’ :’ One Fish, Two Fish, Red Fish, Blue Fish’) }

CODE :

```
object BooksByAuthor {
  def main(args: Array[String]): Unit = {
    val books = List(
      ("Dr. Seuss", "How the Grinch Stole Christmas!"),
```

```
    ("Jon Stone", "Monsters at the End of This Book"),
    ("Dr. Seuss", "The Lorax"),
    ("Jon Stone", "Big Bird in China"),
    ("Dr. Seuss", "One Fish, Two Fish, Red Fish, Blue Fish")
  )

  val countByAuthor = books.groupBy(_._1).mapValues(_.size)

  for ((author, count) <- countByAuthor) {
    println(s"$author has written $count book(s)")
  }
 }
}
```

Output:

```
Jon Stone has written 2 book(s)
Dr. Seuss has written 3 book(s)
```

4.Write the program to illustrate the use of pattern matching in scala, for the following Matching on case classes.

Define two case classes as below:

abstract class Notification

case class Email(sender: String, title: String, body: String) extends Notification case class SMS(caller: String, message: String) extends Notification

Define a function showNotification which takes as a parameter the abstract type Notification and matches on the type of Notification (i.e. it figures out whether it's an Email or SMS).

In the case it's an Email(email, title, _) return the string: s"You got an email from $email wit h title:

$title"

In the case it’s an SMS return the String: s"You got an SMS from $number! Message: $mess
age“

CODE :

```
abstract class Notification


case class Email(sender: String, title: String, body: String) extends Notification

case class SMS(caller: String, message: String) extends Notification


object NotificationApp {

  def showNotification(notification: Notification): String = {

    notification match {

      case Email(sender, title, _) =>

        s"You got an email from $sender with title: $title"

      case SMS(caller, message) =>

        s"You got an SMS from $caller! Message: $message"

    }

  }


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

    val n1 = Email("alice@example.com", "Meeting Reminder", "Don't forget our meeting at
10 AM.")

    val n2 = SMS("123-456-7890", "Call me when you're free.")


    println(showNotification(n1))

    println(showNotification(n2))

  }
```

```
}
```

Output:

```
You got an email from alice@example.com with title: Meeting Reminder
You got an SMS from 123-456-7890! Message: Call me when you're free.
```

5. Write the scala program using imperative style to implement quick sort algorithm.

CODE :

```
object ImperativeQuickSort {
  def quickSort(arr: Array[Int]): Unit = {
    def sort(low: Int, high: Int): Unit = {
      if (low < high) {
        val pivotIndex = partition(low, high)
        sort(low, pivotIndex - 1)
        sort(pivotIndex + 1, high)
      }
    }

    def partition(low: Int, high: Int): Int = {
      val pivot = arr(high)
      var i = low - 1
      for (j <- low until high) {
        if (arr(j) <= pivot) {
          i += 1
          val temp = arr(i)
          arr(i) = arr(j)
          arr(j) = temp
        }
      }
      val temp = arr(i + 1)
      arr(i + 1) = arr(high)
      arr(high) = temp
      i + 1
```

```
    }

    sort(0, arr.length - 1)
  }

  def main(args: Array[String]): Unit = {
    val data = Array(34, 7, 23, 32, 5, 62)
    quickSort(data)
    println("Sorted array: " + data.mkString(", "))
  }
}
```

Output:

```
Sorted array: 5, 7, 23, 32, 34, 62
```

6. Write a scala function to convert the each word to capitalize each word in the given sentence.

```
object CapitalizeWords {
  def capitalizeSentence(sentence: String): String = {
    sentence.split(" ").map(word => word.capitalize).mkString(" ")
  }

  def main(args: Array[String]): Unit = {
    val sentence = "scala is powerful and concise"
    val result = capitalizeSentence(sentence)
    println(result) // Output: Scala Is Powerful And Concise
  }
}
```

Output:

```
Scala Is Powerful And Concise
```

7. Write scala code to show functional style program to implement quick sort algorithm

CODE :

```
object FunctionalQuickSort {
  def quickSort(list: List[Int]): List[Int] = list match {
    case Nil => Nil
    case pivot :: tail =>
      val (less, greater) = tail.partition(_ <= pivot)
      quickSort(less) ::: pivot :: quickSort(greater)
  }

  def main(args: Array[String]): Unit = {
    val data = List(34, 7, 23, 32, 5, 62)
    val sorted = quickSort(data)
    println("Sorted list: " + sorted.mkString(", "))
  }
}
```

Output:

```
Sorted list: 5, 7, 23, 32, 34, 62
```

8. For the below given collection of items with item-names and quantity, write the scala code for the given statement. Items = {(“Pen”:20), (“Pencil”:10), (“Erasor”:7), (“Book”:25), (“Sheet”:15)} i. Display item-name and quantity ii. Display sum of quantity and total number of items iii. Add 3 Books to the collection Add new item “Board” with quantity 15 to the collection I.

CODE :

```
object ItemCollection {
  def main(args: Array[String]): Unit = {
    // Initial collection of items
    var items = Map(
```

```
    "Pen" -> 20,
    "Pencil" -> 10,
    "Erasor" -> 7,
    "Book" -> 25,
    "Sheet" -> 15
  )

  // i. Display item-name and quantity
  println("Item Name and Quantity:")
  items.foreach { case (item, quantity) =>
    println(s"$item: $quantity")
  }

  // ii. Display sum of quantity and total number of items
  val totalQuantity = items.values.sum
  val totalItems = items.size
  println(s"\nTotal Quantity: $totalQuantity")
  println(s"Total Number of Items: $totalItems")

  // iii. Add 3 Books to the collection
  items = items.updated("Book", items("Book") + 3)

  // Add new item "Board" with quantity 15
  items = items.updated("Board", 15)

  // Display updated items
  println("\nUpdated Item Collection:")
  items.foreach { case (item, quantity) =>
    println(s"$item: $quantity")
  }
 }
}
```

Output:

```
Item Name and Quantity:
Erasor: 7
Pencil: 10
Book: 25
Sheet: 15
Pen: 20

Total Quantity: 77
Total Number of Items: 5

Updated Item Collection:
Erasor: 7
Board: 15
Pencil: 10
Book: 28
Sheet: 15
Pen: 20
```

9. Develop a scala code to search an element in the given list of numbers. The function Search() will take two arguments: list of numbers and the number to be searched. The function will write True if the number is found, False otherwise. J.

CODE :

```
object SearchElement {
  def search(list: List[Int], num: Int): Boolean = {
    list.contains(num)
  }

  def main(args: Array[String]): Unit = {
    val numbers = List(3, 7, 10, 2, 8)
    val toFind = 10

    if (search(numbers, toFind)) {
      println("True")
    } else {
      println("False")
    }
  }
```

```
}
```

Output:

```
True
```

10. Illustrate the implementation by writing scala code to generate a down counter from 10 to 1.

CODE :

```
object DownCounter {
  def main(args: Array[String]): Unit = {
    println("Down Counter from 10 to 1:")
    for (i <- 10 to 1 by -1) {
      println(i)
    }}}
```

Output:

```
Down Counter from 10 to 1:
10
9
8
7
6
5
4
3
2
1
```

11. Design a scala function to perform factorial item in the given collection. The arguments passed to the function are the collection of items. Assume the type of the argument for the function suitably. The return type is to be integer.

CODE :

```
object Factorial {
  def factorial(n: Int): Int = {
    if (n == 0) 1
    else n * factorial(n - 1)
  }

  def performFactorial(items: Map[String, Int]): Map[String, Int] = {
    items.map { case (item, quantity) =>
      (item, factorial(quantity))
```

```
  }
 }

 def main(args: Array[String]): Unit = {
  val items = Map("Pen" -> 3, "Pencil" -> 4, "Book" -> 5)
  val result = performFactorial(items)
  result.foreach { case (item, fact) =>
   println(s"$item: $fact")
  }
 }
}
```

Output:

```
Pen: 6
Pencil: 24
Book: 120
```

12. For the below given collection of items with item names and quantity, write the scala code for the given statement. Items = {(“Butter”:20), (“Bun”:10), (“Egg”:7), (“Biscuit”:25),(“Bread”:15)}

i. Display item-name and quantity

ii. Display sum of quantity and total number of items

iii. Add 3 Buns to the collection iv. Add new item “Cheese” with quantity 12 to the collection

CODE :

```
object ItemOperations {

 def main(args: Array[String]): Unit = {
  var items = Map("Butter" -> 20, "Bun" -> 10, "Egg" -> 7, "Biscuit" -> 25, "Bread" -> 15)

  // i. Display item-name and quantity
  println("Item Name and Quantity:")
  items.foreach { case (item, quantity) =>
   println(s"$item: $quantity")
  }

  // ii. Display sum of quantity and total number of items
```

```
    val totalQuantity = items.values.sum
    val totalItems = items.size
    println(s"\nTotal Quantity: $totalQuantity")
    println(s"Total Number of Items: $totalItems")

    // iii. Add 3 Buns to the collection
    items = items.updated("Bun", items("Bun") + 3)

    // iv. Add new item "Cheese" with quantity 12 to the collection
    items = items.updated("Cheese", 12)

    // Display updated items
    println("\nUpdated Item Collection:")
    items.foreach { case (item, quantity) =>
      println(s"$item: $quantity")
    }
  }
}
```

Output:

```
Item Name and Quantity:
Butter: 20
Egg: 7
Bun: 10
Biscuit: 25
Bread: 15

Total Quantity: 77
Total Number of Items: 5

Updated Item Collection:
Butter: 20
Egg: 7
Bun: 13
Cheese: 12
Biscuit: 25
Bread: 15
```

13. Implement function for binary search using recursion in Scala to find the number, given a list of numbers. The function will have two arguments: Sorted list of numbers and the number to be searched.

CODE :

```
object BinarySearch {

  def binarySearch(list: List[Int], low: Int, high: Int, target: Int): Boolean = {

    if (low > high) false

    else {

      val mid = (low + high) / 2

      if (list(mid) == target) true

      else if (list(mid) > target) binarySearch(list, low, mid - 1, target)

      else binarySearch(list, mid + 1, high, target)

    }

  }


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

    val sortedList = List(2, 3, 5, 7, 10, 12, 15, 18)

    val target = 10

    println(binarySearch(sortedList, 0, sortedList.length - 1, target))

  }

}
```

Output:

```
true
```

14. Write a function to find the length of each word and return the word with highest length . Ex for the collection of words = (“games”, “television”,”rope”,”table”) The function should return (“television”,10). The word with the highest length . Read the words from the keyboard.

CODE :

```
object WordLength {

  def longestWord(words: List[String]): (String, Int) = {

    val maxWord = words.maxBy(_.length)

    (maxWord, maxWord.length)

  }
```

```
  def main(args: Array[String]): Unit = {
    val words = List("games", "television", "rope", "table")
    val (word, length) = longestWord(words)
    println(s"Word with highest length: ($word, $length)")
  }
}
```

Output:

```
Word with highest length: (television, 10)
```

15. Define a function f1 that prints "I am function". Write another function f2 that takes a function as a parameter and calls it. Pass f1 to f2 and execute it.

CODE :

```
object FunctionExample {
  def f1(): Unit = {
    println("I am function")
  }

  def f2(func: () => Unit): Unit = {
    func()  // calling the function passed as a parameter
  }

  def main(args: Array[String]): Unit = {
    f2(f1)  // passing f1 to f2
  }
}
```

Output:

```
I am function
```

16. Write a Scala program to demonstrate the use of the following higher-order functions on a list of integers: reduce reduceLeft reduceRight,

CODE :

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

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

    // reduce
    val sum = numbers.reduce(_ + _)
    println(s"Sum using reduce: $sum")

    // reduceLeft
    val leftProduct = numbers.reduceLeft(_ * _)
    println(s"Product using reduceLeft: $leftProduct")

    // reduceRight
    val rightProduct = numbers.reduceRight(_ * _)
    println(s"Product using reduceRight: $rightProduct")
  }
}
```

Output:

```
Sum using reduce: 15
Product using reduceLeft: 120
Product using reduceRight: 120
```