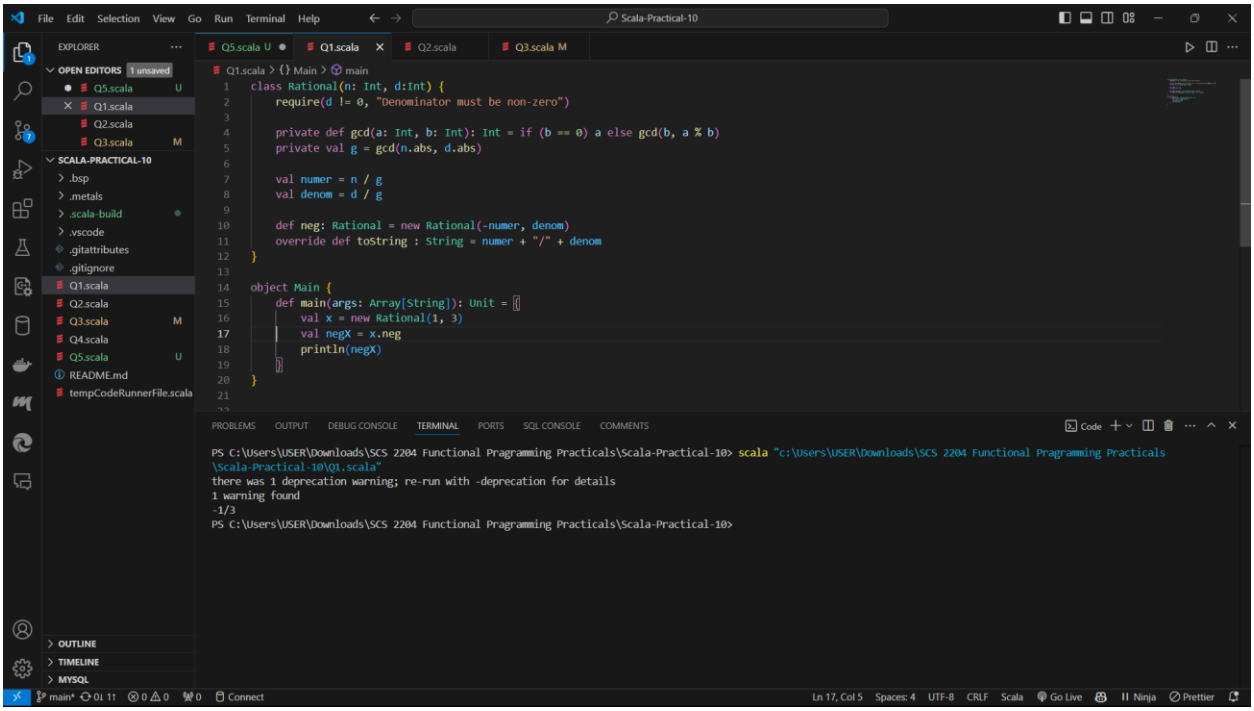
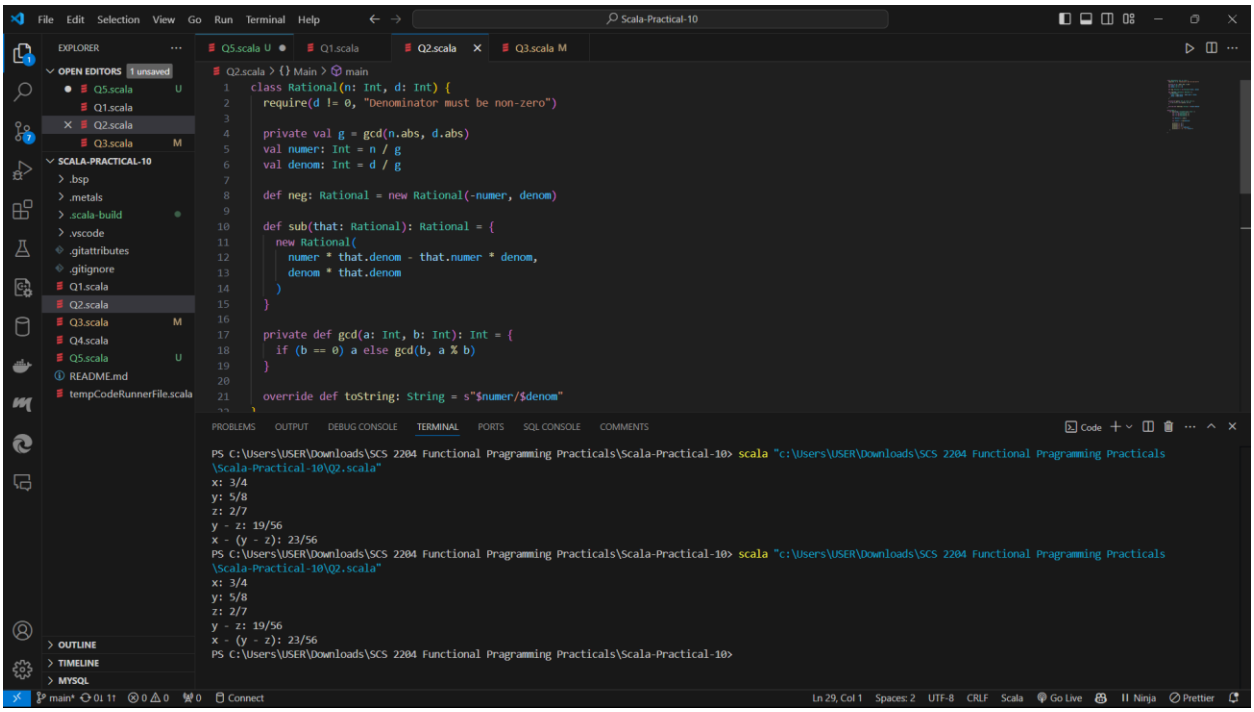
22000862

R.K.J.P.Kashmira

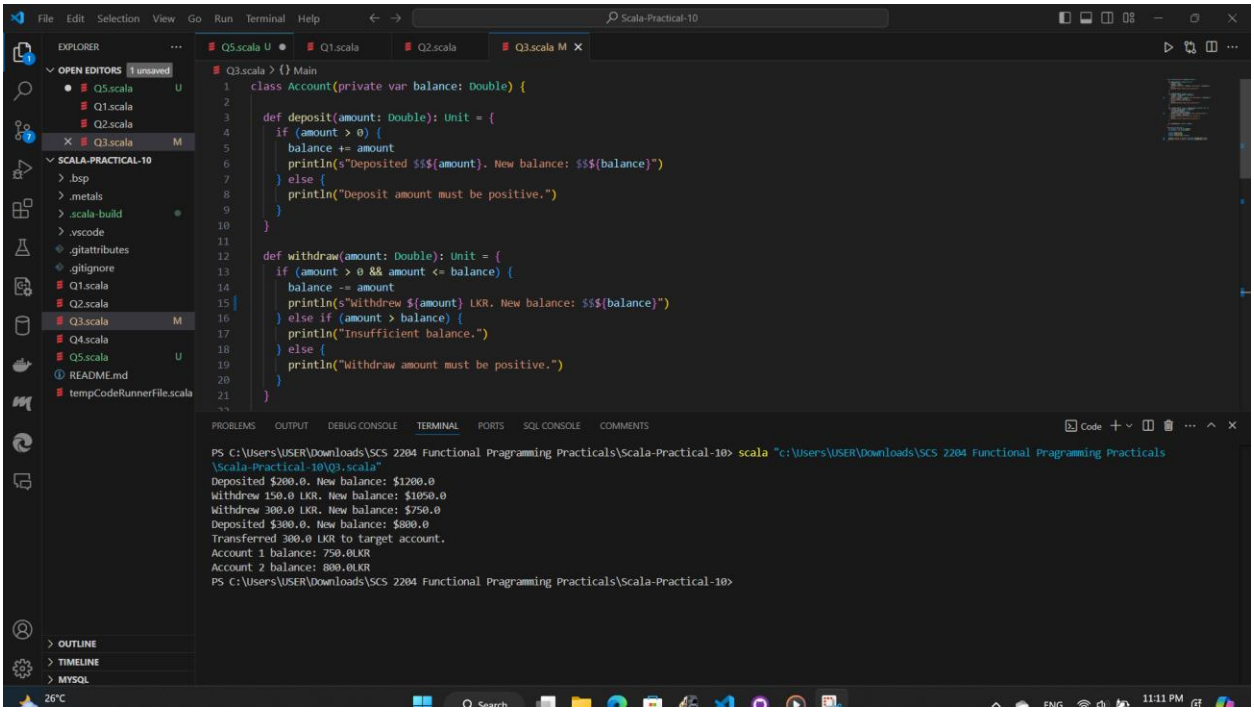
Q1



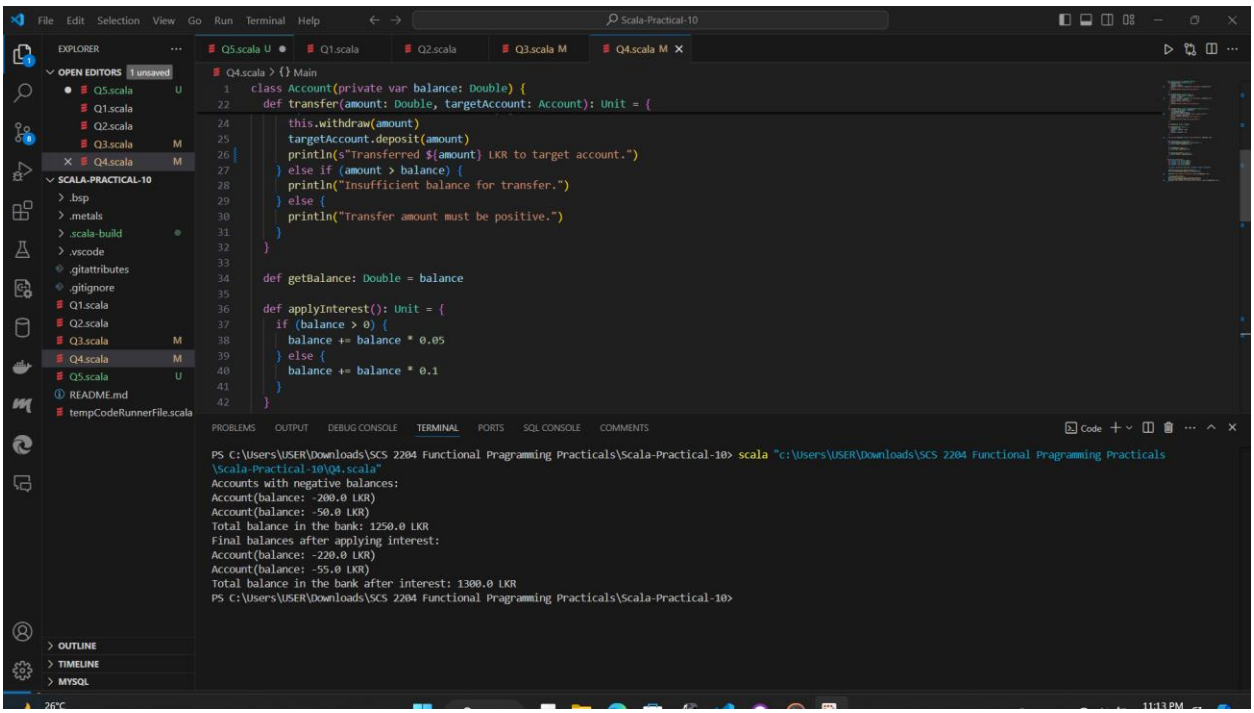
Q2



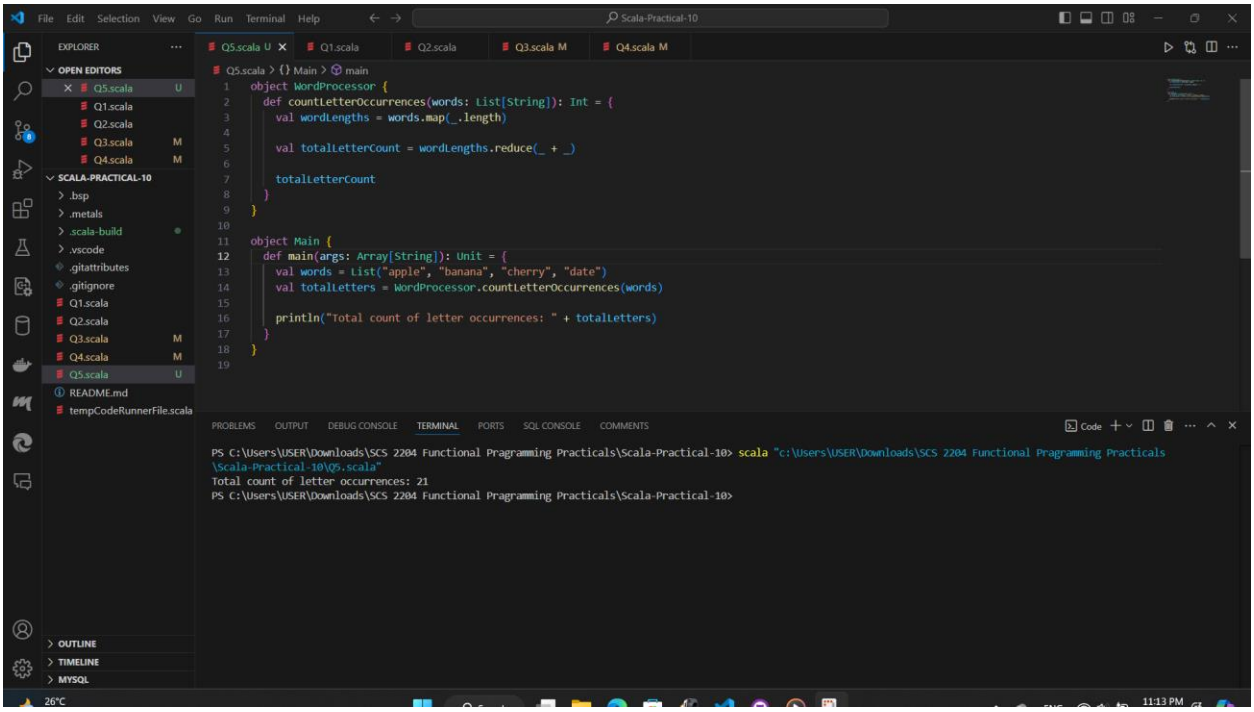
Q3


class Account(private var balance: Double) {
def deposit(amount: Double): Unit = {
if (amount > 0) {
balance += amount
println(s"Deposited $$${amount}. New balance: $$${balance}")
} else {
println("Deposit amount must be positive.")
}
}
def withdraw(amount: Double): Unit = {
if (amount > 0 && amount <= balance) {
balance -= amount
println(s"Withdrew ${amount} LKR. New balance: $$${balance}")
} else if (amount > balance) {
println("Insufficient balance.")
} else {
println("Withdraw amount must be positive.")
}
}
PS C:\Users\USER\Downloads\SCS 2204 Functional Pragramming Practicals\Scala-Practical-10> scala "c:\Users\USER\Downloads\SCS 2204 Functional Pragramming Practicals\Scala-Practical-10\Q3.scala"
Deposited $200.0. New balance: $1200.0
Withdrew 150.0 LKR. New balance: $1050.0
Withdrew 300.0 LKR. New balance: $750.0
Deposited $300.0. New balance: $800.0
Transferred 300.0 LKR to target account.
Account 1 balance: 750.0LKR
Account 2 balance: 800.0LKR
PS C:\Users\USER\Downloads\SCS 2204 Functional Pragramming Practicals\Scala-Practical-10>

Q4


class Account(private var balance: Double) {
def transfer(amount: Double, targetAccount: Account): Unit = {
this.withdraw(amount)
targetAccount.deposit(amount)
println(s"Transferred ${amount} LKR to target account.")
} else if (amount > balance) {
println("Insufficient balance for transfer.")
} else {
println("Transfer amount must be positive.")
}
}
def getBalance: Double = balance
def applyInterest(): Unit = {
if (balance > 0) {
balance += balance * 0.05
} else {
balance += balance * 0.1
}
}
PS C:\Users\USER\Downloads\SCS 2204 Functional Pragramming Practicals\Scala-Practical-10> scala "c:\Users\USER\Downloads\SCS 2204 Functional Pragramming Practicals\Scala-Practical-10\Q4.scala"
Accounts with negative balances:
Account(balance: -200.0 LKR)
Account(balance: -50.0 LKR)
Total balance in the bank: 1250.0 LKR
Final balances after applying interest:
Account(balance: -220.0 LKR)
Account(balance: -55.0 LKR)
Total balance in the bank after interest: 1300.0 LKR
PS C:\Users\USER\Downloads\SCS 2204 Functional Pragramming Practicals\Scala-Practical-10>

Q5


File Edit Selection View Go Run Terminal Help
Scala-Practical-10
EXPLORER
OPEN EDITORS
Q5.scala U
Q1.scala
Q2.scala
Q3.scala M
Q4.scala M
SCALA-PRACTICAL-10
.bsp
.metals
.scala-build
.vscode
.gitattributes
.gitignore
README.md
tempCodeRunnerFile.scala
OUTLINE
TIMELINE
MYSQL
Q5.scala > Main > main
object WordProcessor {
def countLetterOccurrences(words: List[String]): Int = {
val wordLengths = words.map(_.length)
val totalLetterCount = wordLengths.reduce(_ + _)
totalLetterCount
}
}
object Main {
def main(args: Array[String]): Unit = {
val words = List("apple", "banana", "cherry", "date")
val totalLetters = WordProcessor.countLetterOccurrences(words)
println("Total count of letter occurrences: " + totalLetters)
}
}
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL CONSOLE COMMENTS
Code
PS C:\Users\USER\Downloads\SCS 2204 Functional Pragramming Practicals\Scala-Practical-10> scala "c:\Users\USER\Downloads\SCS 2204 Functional Pragramming Practicals\Scala-Practical-10\Q5.scala"
Total count of letter occurrences: 21
PS C:\Users\USER\Downloads\SCS 2204 Functional Pragramming Practicals\Scala-Practical-10>
26°C
11:13 PM