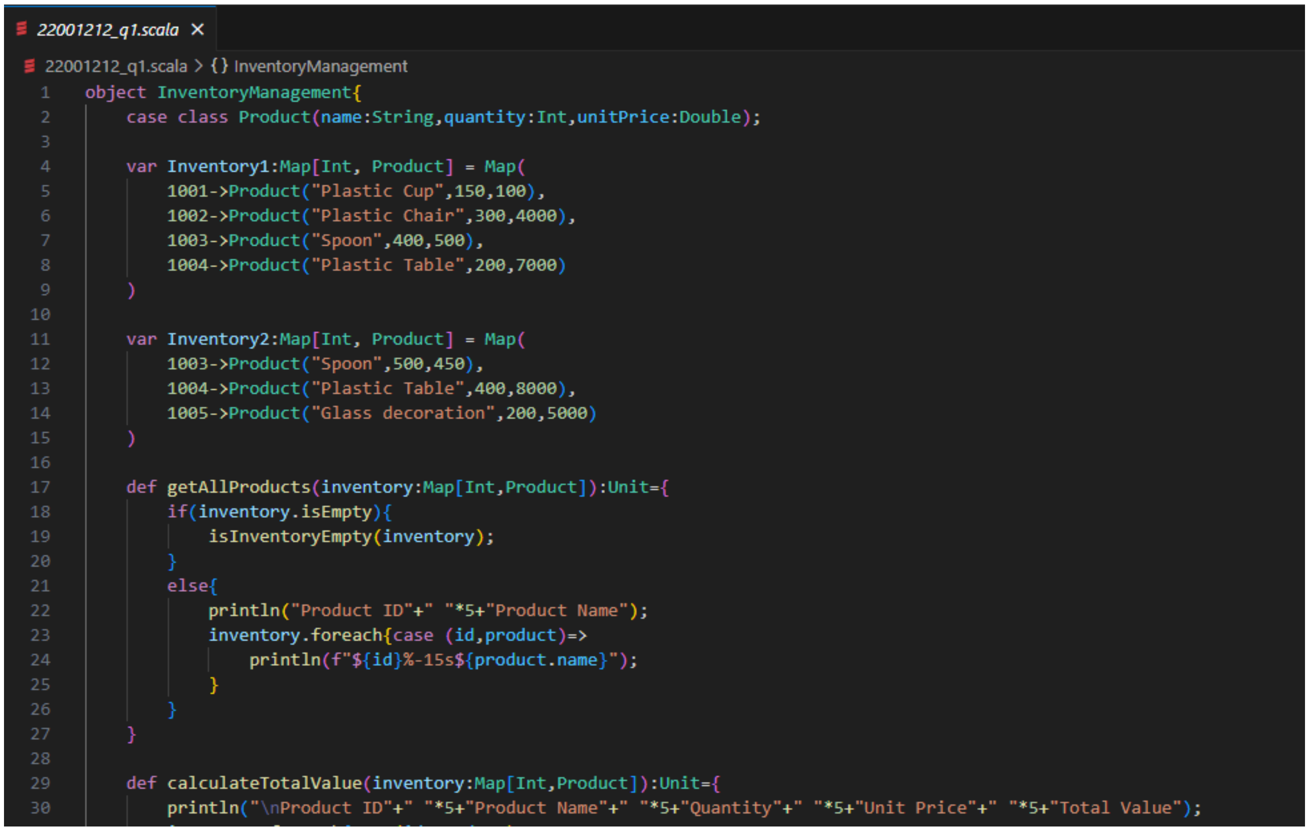
Practical 06 – 22001212

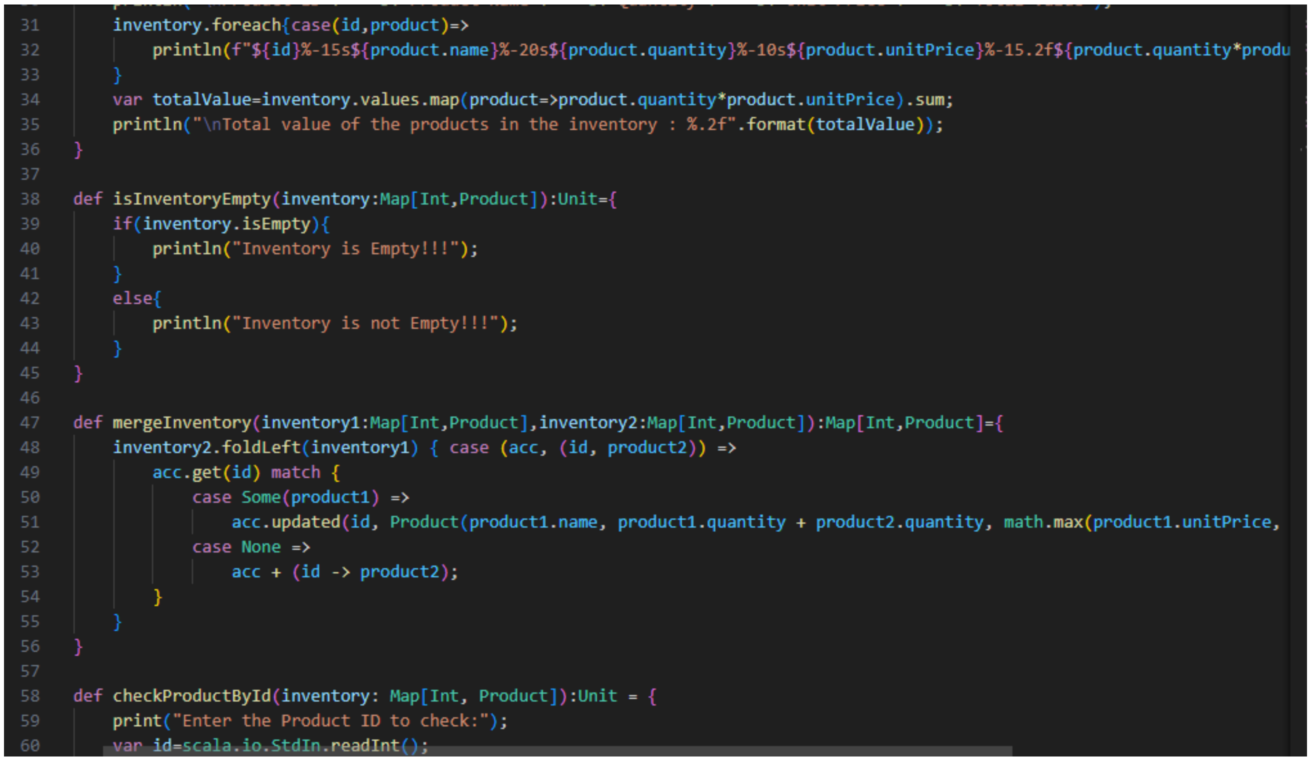
01) Code:



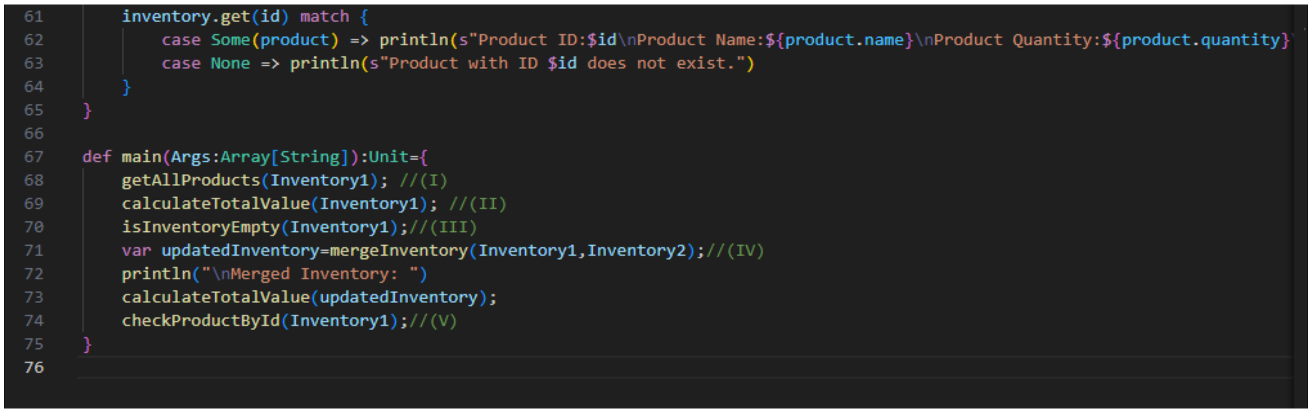
22001212_q1.scala ×

22001212_q1.scala > {} InventoryManagement

```scala
object InventoryManagement{
    case class Product(name:String,quantity:Int,unitPrice:Double);

    var Inventory1:Map[Int, Product] = Map(
        1001->Product("Plastic Cup",150,100),
        1002->Product("Plastic Chair",300,4000),
        1003->Product("Spoon",400,500),
        1004->Product("Plastic Table",200,7000)
    )

    var Inventory2:Map[Int, Product] = Map(
        1003->Product("Spoon",500,450),
        1004->Product("Plastic Table",400,8000),
        1005->Product("Glass decoration",200,5000)
    )

    def getAllProducts(inventory:Map[Int,Product]):Unit={
        if(inventory.isEmpty){
            isInventoryEmpty(inventory);
        }
        else{
            println("Product ID"+" "*5+"Product Name");
            inventory.foreach{case (id,product)=>
                println(f"${id}%-15s${product.name}");
            }
        }
    }

    def calculateTotalValue(inventory:Map[Int,Product]):Unit={
        println("\nProduct ID"+" "*5+"Product Name"+" "*5+"Quantity"+" "*5+"Unit Price"+" "*5+"Total Value");
```

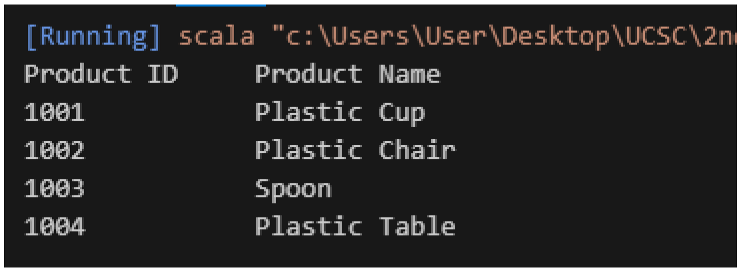


```scala
        inventory.foreach{case(id,product)=>
            println(f"${id}%-15s${product.name}%-20s${product.quantity}%-10s${product.unitPrice}%-15.2f${product.quantity*produ
        }
        var totalValue=inventory.values.map(product=>product.quantity*product.unitPrice).sum;
        println("\nTotal value of the products in the inventory : %.2f".format(totalValue));
    }

    def isInventoryEmpty(inventory:Map[Int,Product]):Unit={
        if(inventory.isEmpty){
            println("Inventory is Empty!!!");
        }
        else{
            println("Inventory is not Empty!!!");
        }
    }

    def mergeInventory(inventory1:Map[Int,Product],inventory2:Map[Int,Product]):Map[Int,Product]={
        inventory2.foldLeft(inventory1) { case (acc, (id, product2)) =>
            acc.get(id) match {
                case Some(product1) =>
                    acc.updated(id, Product(product1.name, product1.quantity + product2.quantity, math.max(product1.unitPrice,
                case None =>
                    acc + (id -> product2);
            }
        }
    }

    def checkProductById(inventory: Map[Int, Product]):Unit = {
        print("Enter the Product ID to check:");
        var id=scala.io.StdIn.readInt();
```



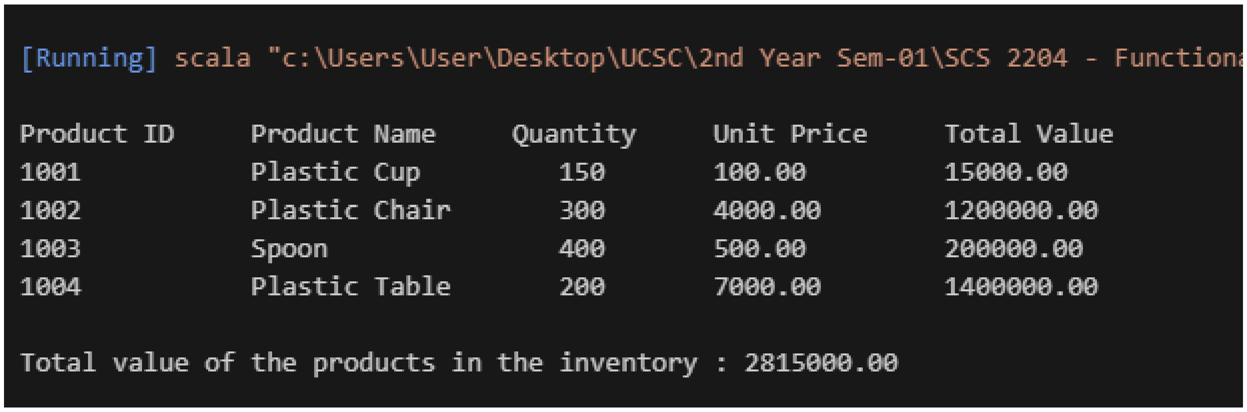
```
    inventory.get(id) match {
        case Some(product) => println(s"Product ID:$id\nProduct Name:${product.name}\nProduct Quantity:${product.quantity}
        case None => println(s"Product with ID $id does not exist.")
    }
}

def main(Args:Array[String]):Unit={
    getAllProducts(Inventory1); //(I)
    calculateTotalValue(Inventory1); //(II)
    isInventoryEmpty(Inventory1);//(III)
    var updatedInventory=mergeInventory(Inventory1,Inventory2);//(IV)
    println("\nMerged Inventory: ")
    calculateTotalValue(updatedInventory);
    checkProductById(Inventory1);//(V)
}
```

I)



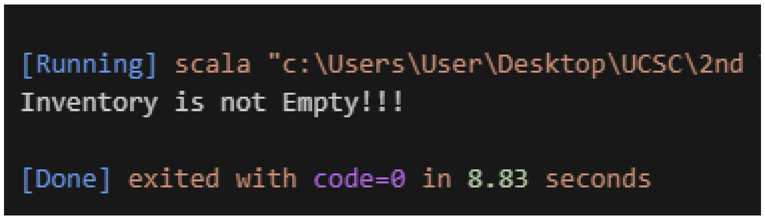
```
[Running] scala "c:\Users\User\Desktop\UCSC\2n
Product ID      Product Name
1001            Plastic Cup
1002            Plastic Chair
1003            Spoon
1004            Plastic Table
```

II)



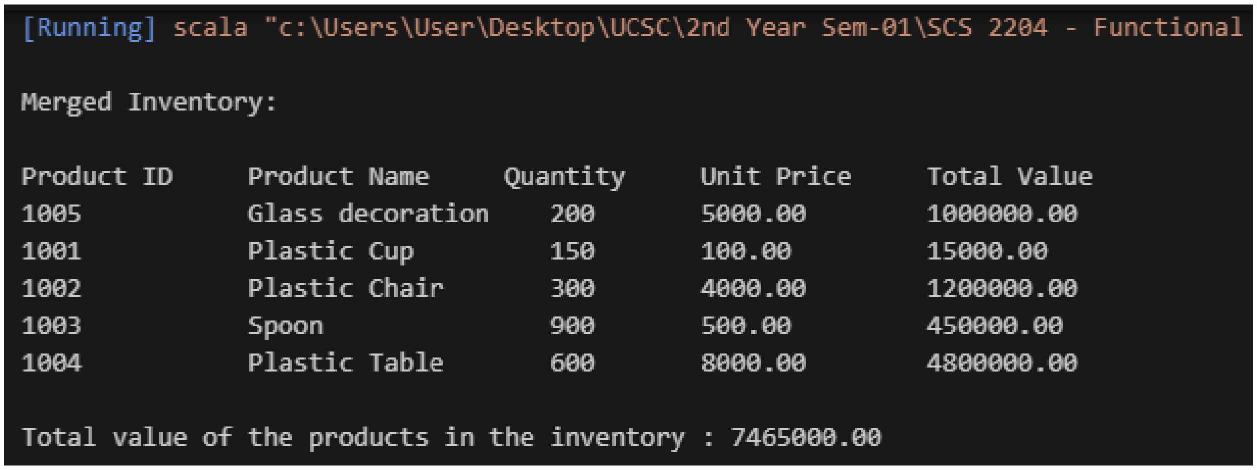
```
[Running] scala "c:\Users\User\Desktop\UCSC\2nd Year Sem-01\SCS 2204 - Functiona

Product ID      Product Name      Quantity      Unit Price      Total Value
1001            Plastic Cup          150        100.00          15000.00
1002            Plastic Chair        300        4000.00         1200000.00
1003            Spoon                400        500.00          200000.00
1004            Plastic Table        200        7000.00         1400000.00

Total value of the products in the inventory : 2815000.00
```

III)



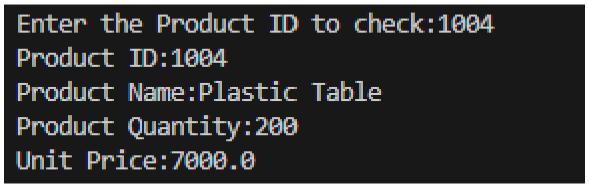
```
[Running] scala "c:\Users\User\Desktop\UCSC\2nd
Inventory is not Empty!!!

[Done] exited with code=0 in 8.83 seconds
```

IV)



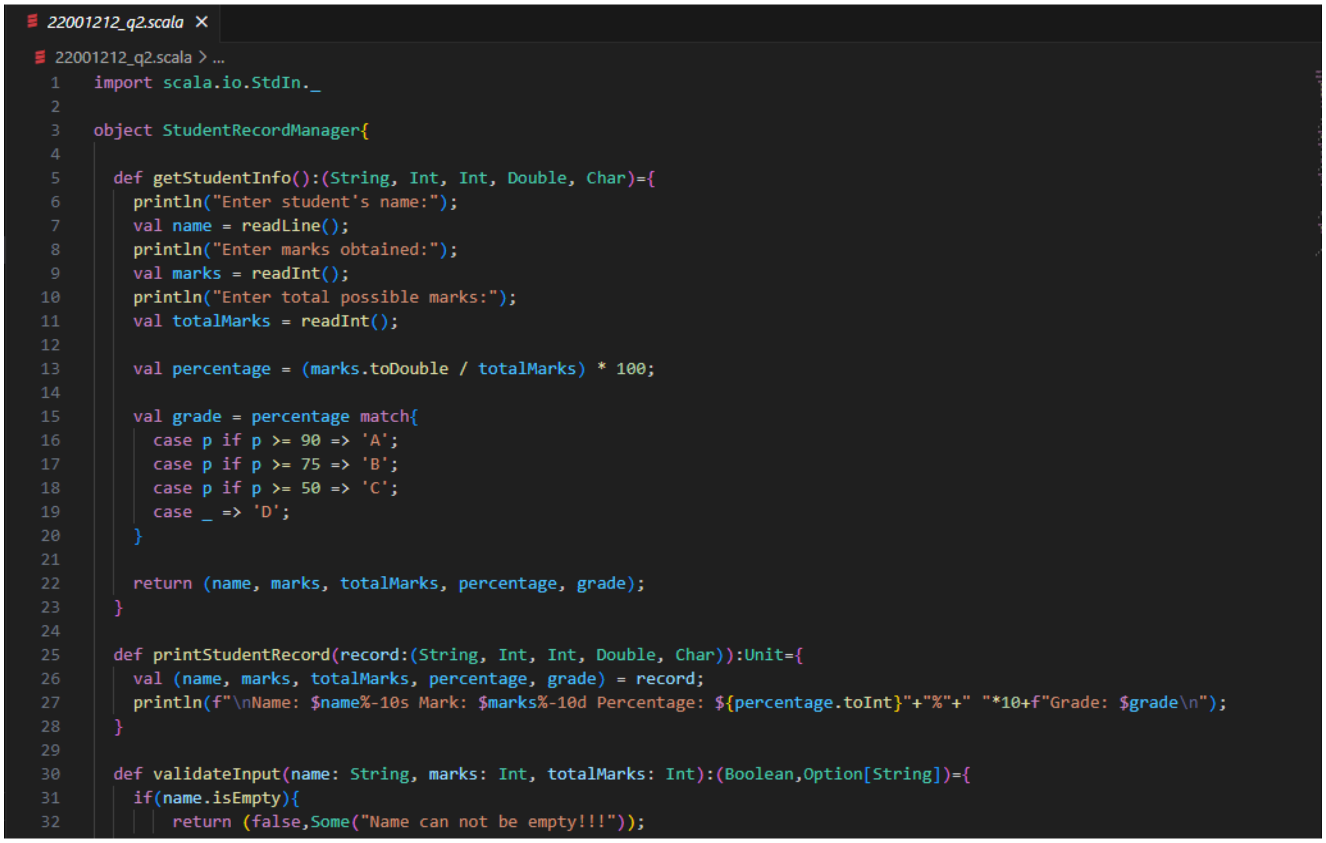
```
[Running] scala "c:\Users\User\Desktop\UCSC\2nd Year Sem-01\SCS 2204 - Functional

Merged Inventory:

Product ID     Product Name      Quantity     Unit Price     Total Value
1005           Glass decoration     200       5000.00        1000000.00
1001           Plastic Cup          150       100.00         15000.00
1002           Plastic Chair        300       4000.00        1200000.00
1003           Spoon                900       500.00         450000.00
1004           Plastic Table        600       8000.00        4800000.00

Total value of the products in the inventory : 7465000.00
```

V)



```
Enter the Product ID to check:1004
Product ID:1004
Product Name:Plastic Table
Product Quantity:200
Unit Price:7000.0
```

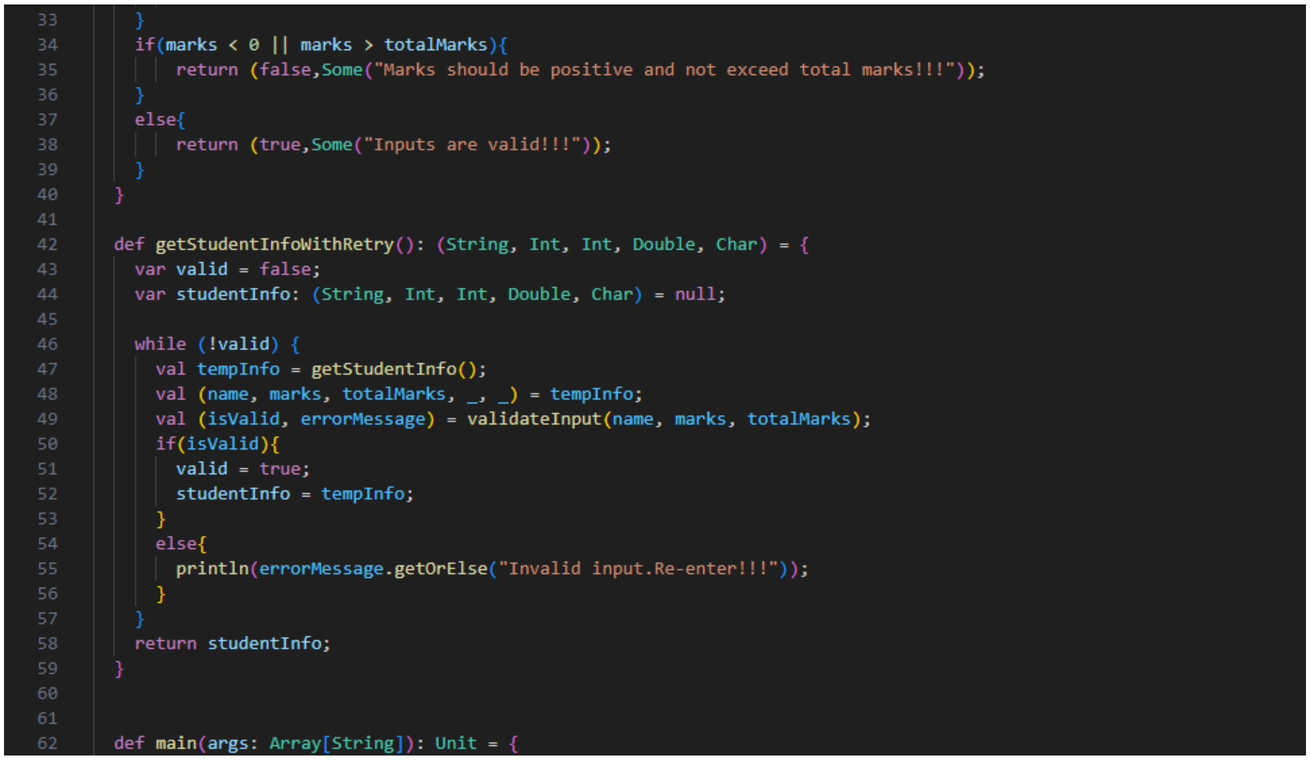
02) Code:



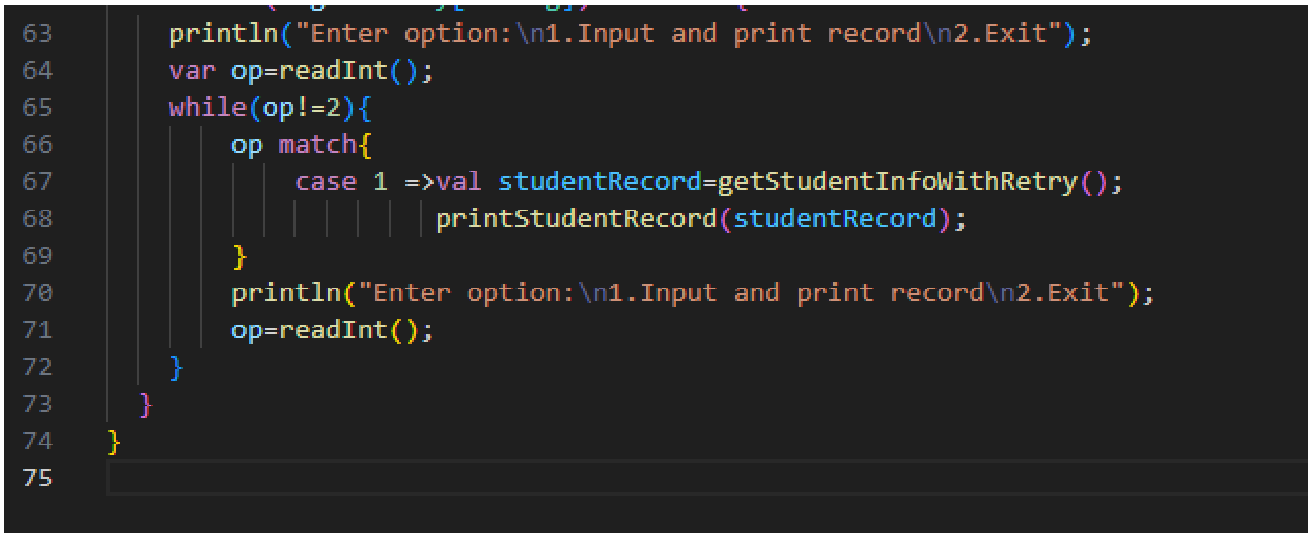
22001212_q2.scala ×

22001212_q2.scala > ...

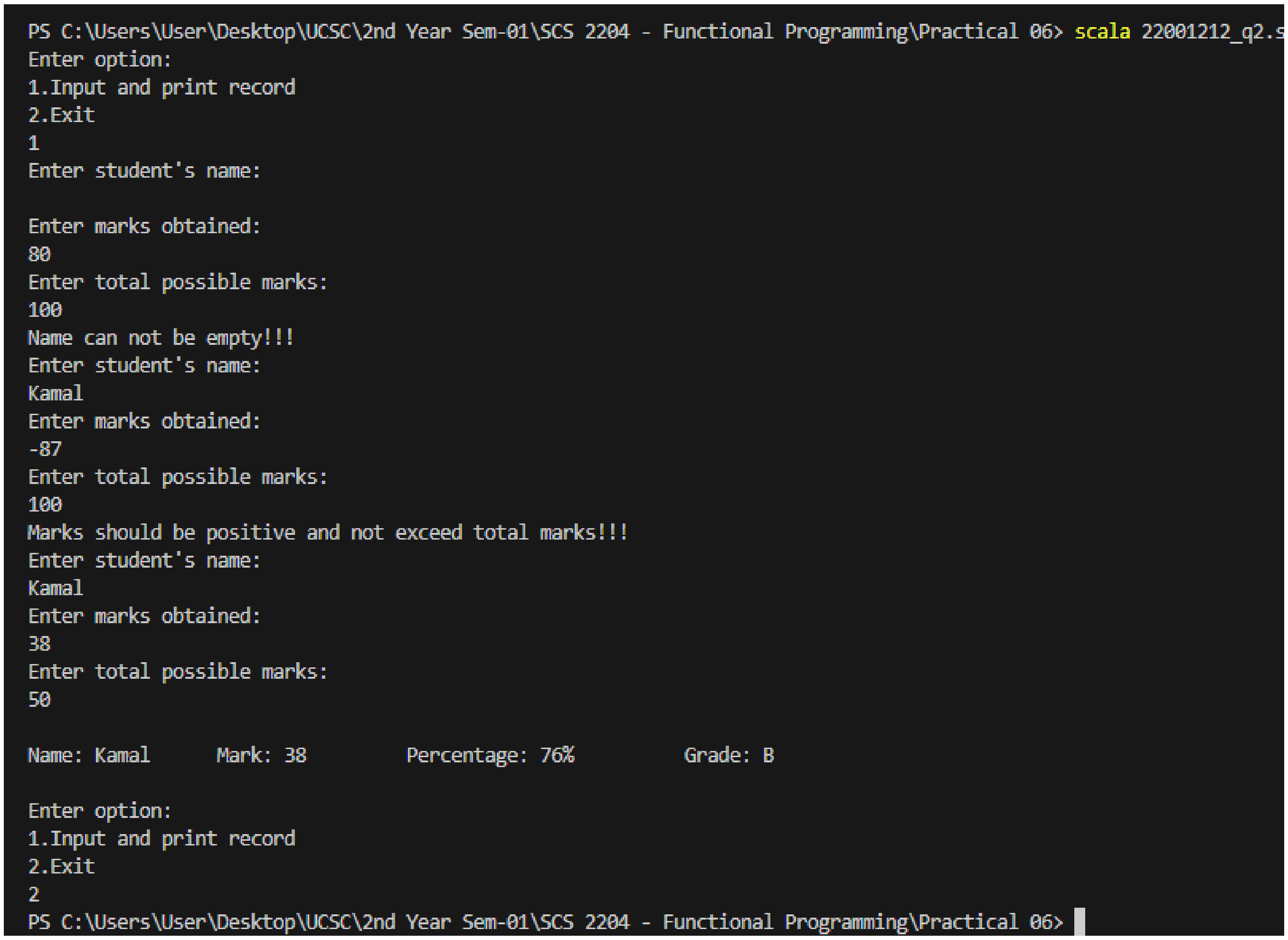
```scala
import scala.io.StdIn._

object StudentRecordManager{

  def getStudentInfo():(String, Int, Int, Double, Char)={
    println("Enter student's name:");
    val name = readLine();
    println("Enter marks obtained:");
    val marks = readInt();
    println("Enter total possible marks:");
    val totalMarks = readInt();

    val percentage = (marks.toDouble / totalMarks) * 100;

    val grade = percentage match{
      case p if p >= 90 => 'A';
      case p if p >= 75 => 'B';
      case p if p >= 50 => 'C';
      case _ => 'D';
    }

    return (name, marks, totalMarks, percentage, grade);
  }

  def printStudentRecord(record:(String, Int, Int, Double, Char)):Unit={
    val (name, marks, totalMarks, percentage, grade) = record;
    println(f"\nName: $name%-10s Mark: $marks%-10d Percentage: ${percentage.toInt}"+"%"+" "*10+f"Grade: $grade\n");
  }

  def validateInput(name: String, marks: Int, totalMarks: Int):(Boolean,Option[String])={
    if(name.isEmpty){
      return (false,Some("Name can not be empty!!!"));
```



```scala
    }
    if(marks < 0 || marks > totalMarks){
        return (false,Some("Marks should be positive and not exceed total marks!!!"));
    }
    else{
        return (true,Some("Inputs are valid!!!"));
    }
  }

  def getStudentInfoWithRetry(): (String, Int, Int, Double, Char) = {
    var valid = false;
    var studentInfo: (String, Int, Int, Double, Char) = null;

    while (!valid) {
      val tempInfo = getStudentInfo();
      val (name, marks, totalMarks, _, _) = tempInfo;
      val (isValid, errorMessage) = validateInput(name, marks, totalMarks);
      if(isValid){
        valid = true;
        studentInfo = tempInfo;
      }
      else{
        println(errorMessage.getOrElse("Invalid input.Re-enter!!!"));
      }
    }
    return studentInfo;
  }


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



```scala
    println("Enter option:\n1.Input and print record\n2.Exit");
    var op=readInt();
    while(op!=2){
        op match{
            case 1 =>val studentRecord=getStudentInfoWithRetry();
                     printStudentRecord(studentRecord);
        }
        println("Enter option:\n1.Input and print record\n2.Exit");
        op=readInt();
    }
  }
}
```



```
PS C:\Users\User\Desktop\UCSC\2nd Year Sem-01\SCS 2204 - Functional Programming\Practical 06> scala 22001212_q2.s
Enter option:
1.Input and print record
2.Exit
1
Enter student's name:

Enter marks obtained:
80
Enter total possible marks:
100
Name can not be empty!!!
Enter student's name:
Kamal
Enter marks obtained:
-87
Enter total possible marks:
100
Marks should be positive and not exceed total marks!!!
Enter student's name:
Kamal
Enter marks obtained:
38
Enter total possible marks:
50

Name: Kamal      Mark: 38         Percentage: 76%          Grade: B

Enter option:
1.Input and print record
2.Exit
2
PS C:\Users\User\Desktop\UCSC\2nd Year Sem-01\SCS 2204 - Functional Programming\Practical 06>
```