SCALA Assignments and solutions

Write a program to create a MAP with empname and deptname. Print details of all employees working in the same department, as “Mr. Joshi”.

```
class Employee(var ename:String,var dept:String)
	{
		def display()
		{
		println("---------------------------");
		println("Name:"+ename);
		println("Department Name:"+dept)
		}
	}
object emp
		{
		def main(args:Array[String])
		{
		val e1=new Employee("Vishnu","finance");
		val e2=new Employee("Sumit","finance");
		val e3= new Employee("Paresh","Marketing");
		val e4 =new Employee("Tushar","Marketing");
		var e5=new Employee("Akshay","Marketing");
		var m1:Map[Int,Employee]=Map(1->e1,2->e2,3->e3,4->e4,5->e5);
		for((k,v)<-m1)
		{
			if(v.dept.equalsIgnoreCase("marketing"))
			v.display()
		}
		}
}
```

Write a program to read five random numbers and convert it to binary and octal using user defined functions.

```
object conv
	{
		def binary(num:Int)
		{
		var bstr=" ";//binary String
		var rem=0;
		println(num);
		var n1=num;
		while(n1>0)
		{
			rem=n1%2;
			n1=n1/2;
```

```
                bstr= rem+bstr;
        }
        println("Binary:"+bstr);
        }

        def octal(num:Int)
        {
                var ostr=" ";//binary String
                var rem=0;
                println();
                println(num);
                var n1=num;
                while(n1>0)
                {
                rem=n1%8;
                n1=n1/8;
                ostr= rem+ostr;
                }
                println("octal:"+ostr);

        }

        def main(args:Array[String])
        {
                val r=new scala.util.Random;
                binary(r.nextInt(15))
                octal(r.nextInt(15))
        }
}
```

Write a program to calculate average of all prime numbers between n1 and n2 (take n1 and n2 from user).

```
object prime
{
        def main(args:Array[String])
        {
        var n1=0;
        var n2=0;
        var count=0;
        var pcount=0;
        var sum=0;
        var prime=" ";
        println("Enter two numbers:");
        n1=scala.io.StdIn.readInt();
        n2=scala.io.StdIn.readInt();
        for(i<-n1 to n2)
        {
```

```
                count=0;
                for(j<-1 to i )
                {
                if(i%j==0)
                {
                count=count+1;
                }
                }
                if(count==2)
                {
                prime=prime+" "+i;
                pcount=pcount+1;
                sum=sum+i;
                }
        }

                println("prime numbers:"+prime);
                println("average:"+sum/pcount);
        }
}
```

Create an abstract class Order (id, description). Derive two classes PurchaseOrder and SalesOrder with details of Supplier and Customer respectively. Create object of each PurchaseOrder And SalesOrder. Display the details of the supplier and customer

```
abstract class Order()
        {
        var orderid:Int=0
        var odescription:String=" ";
        }

class PurchaseOrder( var oid:Int,val descrip:String,var sid:Int,var
sname:String,var pno:Long) extends Order()
{
        orderid=oid;
        odescription=descrip;
        def display()
        {
        println("Order Id:"+orderid);
        println("Description:"+odescription);
        println("Supplier Id:"+sid);
        println("Supplier Name:"+sname);
        println("Phone Number:"+pno);
        }
}
```

```
class SalesOrder(var oid:Int,val descrip:String,var cid:Int,var
cname:String,var pno:Long) extends Order()
{
	orderid=oid;
	odescription=descrip;
	def display()
	{
	println("Order Id:"+orderid);
	println("Description:"+odescription);
	println("Customer Id:"+cid);
	println("Customer Name:"+cname);
	println("Phone Number:"+pno);
	}
}

object sup_ord
{
	def main(args:Array[String])
	{
	var c1=new SalesOrder(1,"TwoLaptops",200,"XYZ",233221);
	var s1=new PurchaseOrder(2,"ThreeComputers",101,"ABC",211231);
	println("Purchase Order");
	println("-----------------------------------------");
	c1.display();
	println("Sales Orders");
	println("-----------------------------------------");
	s1.display();
	}
}
```

Write a program to calculate transpose of a matrix and check if the resultant matrix is lower triangular or not.

```
object transpose_lt
{
	def main(args:Array[String])
	{
		var mat=Array.ofDim[Int](3,3);
		var rmat=Array.ofDim[Int](3,3);
		var isLower:Boolean=true;
		println("Enter Matrix");
		for(i<-0 to 2)
		{
			for(j<-0 to 2)
			{
				mat(i)(j)=readInt();   //scala.io.StdIn.
			}
		}
```

```
		println("Matrix is:");
		for(i<-0 to 2)
		{
			for(j<-0 to 2)
			{
				print(mat(i)(j)+" ");
			}
			println();
		}
		for(i<-0 to 2)
		{
			for(j<-0 to 2)
			{
				rmat(i)(j)=mat(j)(i);
			}
		}
		println("Transepose of Matrix is:");
		for(i<-0 to 2)
		{
			for(j<-0 to 2)
			{
				print(rmat(i)(j)+" ");
			}
			println();
		}
		for(i<-0 to 2)
		{
			for(j<-0 to 2)
			{
				if(i<j)
				{
					if(rmat(i)(j)!=0)
					isLower=false;
				}
			}
		}
	if(isLower==true)
		println("Is Lower Triangular");
	else
		println("Is not Lower Triangular");
	}
}
```

Write a program to create two sets of strings and find common strings between them. Merge sets after removing common strings. Display resultant set.

```
object comm_str
{
def main(args:Array[String])
	{
	var str1:Set[String]=Set("Hello","good","Morning");
	var str2:Set[String]=Set("Hello","good","night");
	var str3=str1.diff(str2);
	println(str1);
	println(str2);
	println(str3);
	var str4=str2.diff(str1);
	println(str4);
	str3++=str4;
	println(str3)
	}
}
```

Write a program to read a character and a string from user and remove first and last occurrence of the character from the string. Display resultant string after reversing its case.

```
object strrev
{
	def reverseString(ch:Char):Char=
	{
		if(ch.isLower)
			ch.toUpper;
		else
			ch.toLower;
	}
	def main(args:Array[String])
	{
		var ch=' ';
		var str=" ";
		println("Enter String:");
		str=scala.io.StdIn.readLine();

		var str1=new StringBuilder(str);
		println("Enter character:");
		ch=scala.io.StdIn.readChar();
		str1.deleteCharAt(str1.indexOf(ch.toString()));
		var str3=str1.deleteCharAt(str1.lastIndexOf(ch.toString())).toString;
		var str4=str3.map(reverseString)
		println(str4);
	}
}
```

Write a program for multiplication of two matrices. Also check if the resultant matrix is upper triangular or not. (Validate number of rows and columns before multiplication and give appropriate message).

```
object Multiplication
{
  def main(args:Array[String])
  {
        println("Enter no of rows and column of 1st matrix")
        var r=readInt()
        var c= readInt()
        println("Enter no of rows and column of 2nd matrix")
        var r1=readInt()
        var c1= readInt()
        var a=Array.ofDim[Int](r,c)
        var b=Array.ofDim[Int](r1,c1)
        var d= Array.ofDim[Int](r,c1)
         if(r1!=c)
        {
             println("Multiplication is not possible")
         }
        else
        {
            println("Enter first matrix")   // to accept
            for(i<-0 to (r-1) ) {
                    for(j<-0 to (c-1)) {
                            a(i)(j)=readInt()
                    }
              }
              println("Enter second matrix")
              for(i<-0 to (r1-1) ) {
                    for(j<-0 to (c1-1)) {
                            b(i)(j)=readInt()
                     }
               }
               println("matrix first")  // to display
               for(i<-0 to (r-1) ) {
                       println()
                       for(j<-0 to (c-1)) {
                            println("\t"+a(i)(j))
                       }
               }
              println("matrix second")
              for(i<-0 to (r1-1) ) {
                 println()
                 for(j<-0 to (c1-1)) {
```

```
                    println("\t"+b(i)(j))
                }
            }
            for(i<-0 to (r-1)) {
                for(j<-0 to (c1-1)) {
                    d(i)(j)=0
                    for(k<-0 to (c-1))
                    {
                        d(i)(j)=d(i)(j)+(a(i)(k)*b(k)(j))
                    }
                }
            }
            println("Multiplication of matrix:-")
            for(i<-0 to (r-1)) {
                println()
                for(j<-0 to (c1-1)) {
                    println("\t" +d(i)(j))
                }
            }
        }//else

        var count=0
        for(i<- 1 to (r-1)) {
            for(j<- 0 to (i-1)) {
                if(d(i)(j) !=0)
                {
                    count=count+1
                }
            }
        }
}
 if(count==0)
 {
        println("matrix is upper triangular")
   }
  else
    {
       println("matrix is not upper triangular")
    }

}//main

 } //object
```

Create array of strings and read a new string from user. Display all the strings from the array that contain the new string.

```
object strdisp
{
	def main(args:Array[String])
	{
	var
str:Array[String]=Array("HelloGoodMorning","HelloGoodNight","HelloGoodAfterno
on")
	var str1=" "
	println("Enter string:")
	str1=readLine()
	var str2=str :+str1
	for(j<-str2)
	{
	println(j)
	}
	}
}
```

Write a program to read two strings. Find the occurrence of second string in the first string. Reverse the case of each occurrence in the string and display resultant string.

```
object occur
{
	def main(args:Array[String])
	{
	val sc=new java.util.Scanner(System.in)  // to take input from cmd
	println("Enter first string")
	val first= sc.nextLine()
	println("Enter second string")
	val second=sc.nextLine()
	if(first.contains(second)==true)
	{
		println("Second string is part of first string")
		/*var n= first.revers(second,null)
		println("After remove second in first string:-\t"+n)*/
	}
	else
	{
		println("Second String is not part of first string")
	}
	}
}
```

Write a program for multiplication of two matrices. Find determinant of resultant matrix.

```
object matmult
{
	def main(args:Array[String])
	{
		val arr1=Array.ofDim[Int](2,2)  //1st array
		val arr2=Array.ofDim[Int](2,2)  //2nd array
		var rarry=Array.ofDim[Int](2,2)  //resultant Array
		println("Enter Matrix1")
		for(i<-0 to 1)
		{
			for(j<-0 to 1)
			{
				arr1(i)(j)=readInt() //read Array1 element
			}
		}
		println("Enter Matrix2")
		for(i<-0 to 1)
		{
			for(j<-0 to 1)
			{
				arr2(i)(j)=readInt() //read Array2 element
			}
		}
		println("MATRIX -1")
		for(i<-0 to 1)
		{
			for(j<-0 to 1)
			{
				print(arr1(i)(j)+" ")  //print Array Element
			}
			println();
		}
		println("MATRIX -2")
		for(i<-0 to 1)
		{
			for(j<-0 to 1)
			{
				print(arr2(i)(j)+" ")  //print Array Element
			}
			println();
		}
		for(i<-0 to 1)
		{
			for(j<-0 to 1)
```

```
                {
                        rarry(i)(j)=0;
                        for(k<-0 to 1)
                                rarry(i)(j)=rarry(i)(j)+arr1(i)(k)*arr2(k)(j)  //multiplication
                }
            }
            println("RESULTANT MATRIX")
            for(i<-0 to 1)
            {
                    for(j<-0 to 1)
                    {
                            print(rarry(i)(j)+" ") //print Array Element
                    }
                    println()
            }
            var det=(rarry(0)(0)*rarry(1)(1))-(rarry(0)(1)*rarry(1)(0))
            println("Determinant:"+det);
        }
}
```

Write a program to merge two sets of integers and calculate sum of all integers in the merged set. Also display largest and smallest element from merged set.

```
import scala.collection.mutable.Set
object setmerge
{
        def main(args:Array[String])
        {
        var s1=Set(1,2,3,4,5,6);
        var s2=Set(4,5,6,7,8);
        s1++=s2;
        println(s1);
        println("Sum:"+s1.sum);
        println("Maximum:"+s1.max);
        println("Minimum:"+s1.min);
        }
}
```

Design an abstract class Employee with computeSal() as an abstract function. Create two subclasses Worker and Manager. Salary of worker should be calculated on hourly basis of work and Salary of Manager should be calculated on monthly basis with additional incentives. Create five objects each of Worker and Manager class, and display their details.

```
abstract class Employee{
```

```
def computesal();
}

class Worker extends Employee
{
var hr:Double = 0.0;
var rate:Double = 0.0;
var tot:Double = 0.0;
def computesal(){

println("Enter no of hour emp work: "); hr= scala.io.StdIn.readDouble();
println("Enter rate per hour of work: "); rate= scala.io.StdIn.readDouble();
tot = hr*rate;

println("Total Salary of worker is : " +tot);

} }

Class Manager extends Employee{
var basic:Double = 0.0;
var hra:Double = 0.0;
var da:Double = 0.0;
var ta:Double = 0.0;
var IT:Double = 0.0;
var PF :Double = 0.0;
var netsal:Double = 0.0;

def computesal(){

println("Enter basic salary of manager: "); basic= scala.io.StdIn.readDouble();
hra = 0.10*basic; ta = 0.08*basic; da = 0.12*basic; IT = 0.15*basic; PF = 0.12*basic;
netsal = (basic+hra+da+ta) - (IT+ PF); println("Salary of Manager is : " + netsal) ;
}

}

object empclass {

def main(args: Array[String]){
        var e1 = new Worker() ;
        var e2 = new Manager() ;
        e1.computesal();
        e2.computesal();
} }
```

Write a program to create a list of 1 to 100 numbers. Create second list from first list selecting numbers which are perfect square. Display it.

```
import scala.collection.mutable.ListBuffer
object list
{
	def main(args:Array[String])
	{
	val l1=List.range(1,101);
	var l2:ListBuffer[Int]=ListBuffer();
	for(i<-l1)
	{
	for(j<-1 to i)
	{
	if(i==j*j)
	l2+=i;
	}
	}
	println("Perfect Numbers:"+l2);
	}
}
```

Write user defined functions to reverse the case of a given string and call the function using MAP.

```
object revstr
{
	def reverse(ch:Char):Char=
	{
	if(ch.isLower)
	ch.toUpper;
	else
	ch.toLower;
	}
	def main(args:Array[String])
	{
	var str=" ";
	println("Enter String:");
	str=readLine();
	var str2=str.map(reverse);
	println(str2);
	}
}
```

Define a class SavingAccount (accNo, name, balance, minBalance). Define appropriate constructors and operations withdraw(), deposit(), viewBalance(). Create an array of SavingAccount objects and perform operations and display them.

```
class SavingAccount(var acno:Int,var name:String,var balance:Int,var minbalance:Int)
{
	def withdraw()
	{
	println("Enter Amount:")
	var n1=readInt()
	balance=balance-n1
	if(balance<minbalance)
	{
	println("TRANSACTION FAILED:")
	balance=balance+n1
	}
	else
	println("TRANSACTION SUCCESSFULL")
	}

def deposite()
{
	println("Enter Amount:")
	var n1=readInt()
	balance=balance+n1;
}

def viewbalance()
{
	println("Account Number:"+acno)
	println("Name:"+name)
	println("Balance:"+balance)
	println("Minimum Balance:"+minbalance)
}
}


object save
{
def main(args:Array[String])
{
val s1=new Array[SavingAccount](5)
var ch=0
s1(0)=new SavingAccount(1,"Akshay Borse",20000,10000)
s1(1)=new SavingAccount(2,"Sumit Amritkar",30000,15000)
s1(2)=new SavingAccount(3,"Vishnu Khatale",40000,6000)
```

```
s1(3)=new SavingAccount(4,"Ganesh Darade",50000,3000)
s1(4)=new SavingAccount(5,"Tushar Amrutkar",55000,10000)
println("Enter Account Number:")
var ac=readInt() //scala.io.StdIn.
for(i<-0 to 4)
{
if(s1(i).acno==ac)
{
println("Account number Exsists")
println("1.Cash Withdraw:")
println("2.Cash Deposite:")
println("3.View Balance:")
println("4.Exit")
while(ch!=5)
{
println("Enter YourChoice:")
var
ch=readInt()
ch match
{
case
1=>s1(i).withdraw()
case
2=>s1(i).deposite()
case
3=>s1(i).viewbalance()
case
4=>System.exit(1)
}
}
}
}
println()
}
}
```

Write a program to calculate sum of all perfect numbers between 1 and 100. Display perfect numbers also.

```
object perf
{
        def main(args:Array[String])
        {
        var sum=0;
        var psum=0;
```

```
	var perfect=" ";
	for(i<-1 to 100)
	{
		for(j<-1 to i-1)
	{
	if(i%j==0)
		{
		sum=sum+j;
		}
	}
	if(sum==i)
	{
		psum=psum+i;//sum of perfect number;
		perfect=perfect+" "+i;
	}
	sum=0;
	}
	println("perfectNumbers:"+perfect);
	println("Sum of Perfect Number:"+psum);
	}
}
```

Create lists using five different methods and display each of them.(List style, java style, fill, range, tabulate methods)

```
object list1
{
	def main(args:Array[String])
	{
		//For creating in lisp style
		val list=10::20::30::Nil
		println("Lisp style is:-"+list)

		//For creating in Java style
		val list2=List(11,22,33)
		println("Java style is:-"+list2)

		//For creating in fill
		val x3=List.fill(3)("food")  //  it will print food 3 times
		println("Fill is:-"+x3)

		//For creating in range
		val x1=List.range(1,10)
		println("Range is:-"+x1)  // it will print from 1 to 9
```

```
		//For creating tabulate method
		val x4=List.tabulate(5)(n=>n*n)  // it will print square of nos from 0 to 4
		println("Tabulated Format is:-"+x4)
	}
}
```

Create a list of 50 members using function 2n+3. Create second list excluding all elements multiple of 5.

```
object fneq
{
	def main(args:Array[String])
	{
		val l=List.tabulate(50)(n=>(2*n)+3)
		println(l)

		val s=List.tabulate(50)(n=>(2*n)+3).filterNot(i=>i%5==0)
		println("Excludes of multiple 7:-"+s)
	}
}
```

Create a list of 10 random numbers. Create another list from members of first list using function 3n2+4n+6. Display second list in ascending order.

```
import scala.util._
object s20
{
	def main(args:Array[String])
	{
	var l1:List[Int]=List();
	var l2:List[Int]=List();
	var n1=0;
	for(i<- 1 to 10)
	{
	l1::=Random.nextInt(10);
	}
	println("List1:"+l1.sorted);
	for(j<-l1)
	{
	n1=3*j*j+4*j+6;
	l2::=n1;
	}
	println("list2:"+l2.sorted)
	}
}
```