FY MCS SCALA PRACTICAL SOLVED SLIPS

SLIP 1-Q1

Write a program for multiplication of two matrices. Find determinant of resultant matrix. (Accept matrix: 5M Multiplication : 10M Determinant : 10M Display : 5M)

object Sample {

def main(args: Array[String]) {

var Matrix1 = Array.ofDim[Int](2, 2)

var Matrix2 = Array.ofDim[Int](2, 2)

var Matrix3 = Array.ofDim[Int](2, 2)

var i: Int = 0

var j: Int = 0

var k: Int = 0

var sum: Int = 0

printf("Enter elements of MATRIX1:\n")

i = 0;

while (i < 2) {

j = 0;

while (j < 2) {

printf("ELEMENT(%d)(%d): ", i, j);

Matrix1(i)(j) = scala.io.StdIn.readInt();

j = j + 1;

}

i = i + 1;

}

printf("Enter elements of MATRIX2:\n")

i = 0;

while (i < 2) {

j = 0;

while (j < 2) {

printf("ELEMENT(%d)(%d): ", i, j);

Matrix2(i)(j) = scala.io.StdIn.readInt();

j = j + 1;

}

i = i + 1;

}

//Multiply Matrix1 and Matrix2

i = 0;

while (i < 2) {

j = 0;

while (j < 2) {

sum = 0;

k = 0;

while (k < 2) {

sum = sum + (Matrix1(i)(k) \* Matrix2(k)(j));

k = k + 1;

}

Matrix3(i)(j) = sum;

j = j + 1;

}

i = i + 1;

}

printf("MATRIX1:\n")

i = 0;

while (i < 2) {

j = 0;

while (j < 2) {

printf("%d ", Matrix1(i)(j));

j = j + 1;

}

i = i + 1;

println();

}

printf("MATRIX2:\n")

i = 0;

while (i < 2) {

j = 0;

while (j < 2) {

printf("%d ", Matrix2(i)(j));

j = j + 1;

}

i = i + 1;

println();

}

printf("Multiplication of Matrix1 and Matrix2:\n")

i = 0;

while (i < 2) {

j = 0;

while (j < 2) {

printf("%d ", Matrix3(i)(j));

j = j + 1;

}

i = i + 1;

println();

}

}

}

SLIP 2-Q1

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. (read : 5M find occurance:10M reverse : 10M display: 5M)

import scala.io.StdIn.\_

import scala.io.StdIn.{readLine}

object str2

{

def main(args: Array[String])

{

println("Enter the first string:");

var str1=readLine()

println("Enter second String:")

var str2=readLine()

if(str1.contains(str2))

println("second string is present in first string")

else

println("not present")

var newstr = str2.reverse

println("string after reversing the occurence:"+newstr)

}

}

SLIP3-Q1.[A]

Write a program to read five random numbers and convert it

to binary and octal using user defined functions.

(random number : 5M binary : 5M Octal : 5M)

B. Write a program to calculate average of all prime numbers

between n1 and n2 (take n1 and n2 from user).

(accept n1, n2 : 5M prime numbers : 5M average :5M)

object slip2a

{

    def binary(n:Int):String=

    {

            var n1:Int=n

            var s:String=""

            while(n1>0)

            {

                s=s.concat((n1%2).toString())

                n1= n1/2

            }

            return  s.reverse

    }

    def octal(n:Int):String=

    {

        var n1:Int=n

        var s:String=""

        while(n1>0)

        {

            s=s.concat((n1%8).toString())

            n1= n1/8

        }

        return s.reverse

    }

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

    {

        println("Enter 5 random numbers:")

        var rand=scala.util.Random

        var arr:Array[Int]=new Array[Int](5)

        for(i<- 0 to 4)

        {

            arr(i)=rand.nextInt(100)

            println("Number "+i+" of "+arr(i)+" in binary is:"+binary(arr(i)))

        println("Number "+i+" of "+arr(i)+" in octal is:"+octal(arr(i)))

        }

        /\*println("enter a number:")

        var n=scala.io.StdIn.readInt()

        println(binary(n))

        println(octal(n))

        \*/

    }

}

SLIP3-(B)

object slip2b

{

    def isprime(n:Int):Boolean=

    {

            for(i<-2 to n/2)

            {

                if(n%i==0) return false

            }

            return true

    }

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

    {

        println("Enter n1:")

        var n1:Int=scala.io.StdIn.readInt()

        println("Enter n2:")

        var n2:Int=scala.io.StdIn.readInt()

        var sum=0

        var cnt=0

        for(i<- n1 to n2)

        {

            if(isprime(i))

            {

                sum+=i

                cnt+=1

            }

        }

        println("count is:"+cnt)

        var result=sum/cnt

        println("sum of prime numbers between "+n1+" and "+n2+" is:"+sum)

        printf("Average of prime numbers between "+n1+" and "+n2+" is:"+result)

    }

}

SLIP 4- Q1

Create an abstract class Order (id, description). Derive two classes

PurchaseOrder and SalesOrder with details of Supplier and Customer.

Create object of each PurchaseOrder And SalesOrder. Display the

details of all supplier and customers.

(abstract class : 5M derivation : 20 M display : 10M)

import scala.annotation.switch

abstract class order(id1:Int,desc:String)

{

    var id=id1

    var description=desc

    def orderdetails()

}

class purchase(id1:Int,desc:String,v:String,c:String)extends order(id1:Int,desc:String)

{

    var vendor=v

    var customer=c

    def orderdetails()

    {

        println("\_\_\_\_\_\_\_\_\_\_\_\_\_\*\*PURCHASE\*\*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

        println("order id"+id)

        println("order descreption"+description)

        println("vendor"+vendor)

        println("customer"+customer)

    }

}

class sales(id1:Int,desc:String,v:String,c:String)extends order(id1:Int,desc:String)

{

    var vendor=v

    var customer=c

    def orderdetails()

    {

        println("\_\_\_\_\_\_\_\_\_\_\_\_\_\*\*SALES\*\*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

        println("order id"+id)

        println("order descreption"+description)

        println("vendor"+vendor)

        println("customer"+customer)

    }

}

object order

{

    def main(args:Array[String])

    {

        while(true)

        {

            println("Enter your choices\n1:purchase\n2:sales\n3:Exit")

            val ch:Int=readInt()

            ch match

            {

                case 1=>

                    println("order Id:")

                    var id=readInt()

                    println("order descreiption:")

                    var desc=readLine()

                    println("Vendor:")

                    var v=readLine()

                    println("Customer")

                    var c=readLine()

                    var p=new purchase(id,desc,v,c)

                    p.orderdetails()

                case 2=>

                    println("order Id:")

                    var id=readInt()

                    println("order descreiption:")

                    var desc=readLine()

                    println("Vendor:")

                    var v=readLine()

                    println("Customer")

                    var c=readLine()

                    var  s=new sales(id,desc,v,c)

                    s.orderdetails()

                case 3=>

                    System.exit(1)

            }

        }

    }

}

SLIP-5 Q1

Write a program to create a MAP with empno, empname and deptname. Print details of all employees working in the same department, as “Mr. Joshi” . (create MAP : 10M logic: 10M print :5M)

import scala.io.StdIn.{readLine,readInt}

import scala.collection.mutable.Map

object Map\_p2{

def main(args:Array[String]){

println("Enter no of employees")

var n=readInt()

var map:Map[String,String]=Map[String,String]()

var empname:String=null

var deptname:String=null

println("Enter " + n + " employee details:")

var i=1

for(i<-1 to n){

println("Enter employee name: ")

empname=readLine()

println("Enter department")

deptname=readLine()

map+=(empname->deptname)

}

println(map)

println("Empolyee with Same department as Joshi(" + map("Joshi") + "):")

var j = 1

for(i<-map.keys)

{

if(map(i) == map("Joshi"))

{

println(j+". "+i)

j+=1

}

}

}

}

SLIP 6 Q1

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.

object Scala\_List

{

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

{

//Iterate over a list

val nums = List(1, 3, 5, 7, 9, 11, 14, 12)

println("Original list:")

println(nums)

println("Largest number of the said list:")

println(nums.max)

println("Smallest number from the said list:")

println(nums.min)

}

}

SLIP 7- Q1

Create Lists using five different methods

( Lisp style , Java style, fill, range and tabulate methods).\*/

object Scala\_List

{

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

{

println("Scala List:")

println("Lisp style:")

val lisp\_list = 101 :: 226 :: 30 :: Nil

println(lisp\_list)

println("Java style:")

val num = List(1,2,3,4,5,6,7)

println(num)

println("Range List:")

val b = List.range(1, 15)

println(b)

val c = List.range(0, 20, 2)

println(c)

println("Uniform List:")

val d = List.fill(3)("PPL")

println(d)

println("Tabulated List:")

val e = List.tabulate(5)(n => n \* n \* n)

println(e)

}

}

SLIP 8 -Q1

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

object

{

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;

perfect=perfect+" "+i;

}

sum=0;

}

println("perfectNumbers:"+perfect);

println("Sum of Perfect Number:"+psum);

}

}

SLIP 9 – Q1

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

object slip4

{

    def transposemtr(matrix:Array[Array[Int]],m:Int,n:Int):Array[Array[Int]]=

    {

        var mtr=Array.ofDim[Int](n,m)

        for(i <- 0 until m ; j<- 0 until n)

        {

            mtr(i)(j)=matrix(j)(i)

        }

        return mtr

    }

    def lowertriangular(matrix:Array[Array[Int]],m:Int,n:Int):Boolean=

    {

        for(i <- 0 until m; j <- 0 until n)

        {

            if(i<j)

            {

                if(matrix(i)(j)!=0) return false

            }

        }

        return true

    }

    def printmatrix(matrix:Array[Array[Int]],m:Int,n:Int):Unit=

    {

        for(i <- 0 until m)

        {

            println()

            for(j <- 0 until n)

            {

                print(matrix(i)(j)+ " ")

            }

        }

    }

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

    {

        println("How many rows you want to enter in matrix 1:")

        var m:Int=scala.io.StdIn.readInt()

        println("How many columns you want to enter in matrix 1:")

        var n:Int=scala.io.StdIn.readInt()

        println("Enter elements:")

        var matrix=Array.ofDim[Int](m,n)

        for(i<- 0 until m ; j <- 0 until n)

        {

            matrix(i)(j)=scala.io.StdIn.readInt()

        }

        var ans=transposemtr(matrix,m,n)

        println("Transpose is:" )

        printmatrix(ans,m,n)

         if(lowertriangular(ans,m,n))

         {

            println("transposed matrix is lower triangular.")

         }

         else println("not lower triangular")

    }

}

SLIP 10 –Q1

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 rev

{

    def convert(c:Char):Char=

    {

        if(c.isLower)

            return (c.toUpper)

        else

            return (c.toLower)

    }

    def main(args: Array[String])

    {

        println("Enter String")

        var str=scala.io.StdIn.readLine()

        println("String After reversing the case:")

        println(str.map(convert))

    }

}

SLIP 11- Q1

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 and function :10M subclasses:5M salary calculation: 5 M each display:5M)

import scala.annotation.switch

abstract class Employee(id:Int,name:String)

{

    var id1=id

    var name1=name

    def computeSal()

    def disp()

}

class Worker(id:Int,name:String,Wid:Int,Wname:String,hours:Int,Rate:Int) extends Employee(id,name)

{

    var sal=0

    var Wid1=Wid

    var Wname1=Wname

    var hours1=hours

    var Rate1=Rate

    def computeSal()

    {

        sal=Rate\*hours

    }

    def disp()

    {

        println("\n\nWorker ID:"+Wid)

        println("Worker Name:"+Wname)

        println("Number of hours worked:"+hours)

        println("(Rate/Hour):"+Rate)

        println("Worker Salary is:"+sal)

    }

}

class Manager(id:Int,name:String,Mid:Int,Mname:String,salary:Int,incent:Int) extends Employee(id,name)

{

    var sall=0

    var Mid1=Mid

    var Mname1=Mname

    var salary1=salary

    var incent1=incent

    def computeSal()

    {

        sall=salary+incent

    }

    def disp()

    {

        println("Manager ID:"+Mid)

        println("Manager Name:"+Mname)

        println("Manager Monthly Salary:"+salary)

        println("Manager Incentives:"+incent)

        println("\n Manager Total Salary is:"+sall)

    }

}

object Employee

{

    def main(arg:Array[String])

    {

        println("\nEnter Employee ID:")

        var id=readInt()

        println("\nEnter Employee Name:")

        var name=readLine()

        println("\n Employee Id:"+id)

        println("\n Employee Name:"+name)

        while(true)

            {

                println("Enter your choice:\n1:Worker\n2.Manager\n3:Exit")

                val ch:Int=readInt()

                ch match

                {

                    case 1=>

                        println("Enter Worker ID:")

                        var Wid=readInt()

                        println("Enter Worker Name:")

                        var Wname=readLine()

                        println("Enter Number of hours worked:")

                        var hours=readInt()

                        println("Enter (Rate/Hour):")

                        var Rate=readInt()

                        var wo=new Worker(id,name,Wid,Wname,hours,Rate)

                        wo.computeSal()

                        wo.disp()

                    case 2=>

                        println("Enter Manager ID:")

                        var Mid=readInt()

                        println("Enter Manager Name:")

                        var Mname=readLine()

                        println("Enter Manager Salary:")

                        var salary=readInt()

                        println("Enter Manager Incentives:")

                        var incent=readInt()

                        var ma=new Manager(id,name,Mid,Mname,salary,incent)

                        ma.computeSal()

                        ma.disp()

                    case 3=>System.exit(1)

                }

            }

        }

}

SLIP 12 – Q1

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. (create first list: 10M find perfect square : 10M create second list :5 M display : 5M)

object List5 {

def main(args: Array[String]) {

val list1 = List.tabulate(50)(n => 2\*n+3 )

println( "list1 : " + list1 )

val list2 = List.tabulate(5)(n => n \* 5)

println("list2: "+list2)

}

}

SLIP 13 – Q1

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

// Creating object

object GfG

{

// Main method

def main(args:Array[String])

{

// Creating a map

val m1 = Map(3 -> "geeks", 4 -> "for", 2 -> "cs")

// reversing key:value pairs

val reverse = for ((k, v) <- m1) yield (v, k)

// Displays output

println(reverse)

}

}

SLIP 14 Q1

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 slip5

{

    def main(args:Array[String])

    {

        var set1=scala.collection.mutable.Set[Int]()

        var set2=scala.collection.mutable.Set[Int]()

        var ele=0

        println("Enter number of elementsin set1:")

        var n:Int=scala.io.StdIn.readInt()

        println("Enter elements in set1:")

        for(i <- 1 to n)

        {

            ele=scala.io.StdIn.readInt()

            set1+=ele

        }

        println("Enter number of elementsin set2:")

        n=scala.io.StdIn.readInt()

        println("Enter elements in set2:")

        for(i <- 1 to n)

        {

            ele=scala.io.StdIn.readInt()

            set2+=ele

        }

        println("elements is set1 are:"+set1)

        println("elements in set2 are:"+set2)

        var common= set1.intersect(set2)

        println("Common elements are:"+common)

        var set5=set1.union(set2)&~(common)

        println("After removal of common string the set is:"+set5)

    }

}

SLIP 15 Q1

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.

(define class:5M constructors: 10M operations: 5 M each)

class SavingAcc(ano:Int,n:String,bal:Int)

{

    var accNo=ano

    var name=n

    var balance=bal

    var minbalance:Int=0

    def this(ano:Int,n:String,b:Int,mb:Int)

    {

        this(ano,n,b)

        this.minbalance=mb

    }

    def viewBal()

    {

        println("Account Num:"+accNo)

        println("Owner name:"+name)

        println("Balance:"+balance)

    }

    def withdraw()

    {

        println("Enter amount to withdraw")

        var wamt=readInt()

        if(balance-wamt>minbalance)

        {

            balance-=wamt

            println("Remaining balance:"+balance)

        }

        else

            println("Insufficient Fund In Account")

    }

    def deposite()

    {

        println("Amount to deposite:")

        val damt=readInt()

        balance+=damt

        println("New Balance:"+balance)

    }

}

object account

{

    def main(args:Array[String])

    {

        println("New account?1/0")

        var ch=readInt()

        println("\*\*\*\*\*:"+ch)

        while(ch==1)

        {

            println("Enter account number:")

            val ano=readInt()

            println("Enter Acount Owner name:")

            var name=readLine()

            println("Enter account balance")

            var bal=readInt()

            var newacc=new SavingAcc(ano,name,bal,500)

            newacc.viewBal()

            newacc.deposite()

            newacc.viewBal()

            newacc.withdraw()

            newacc.viewBal()

            println("New account?1/0")

            var ch=readInt()

            if(ch==0)

                System.exit(0)

        }

    }

}

SLIP 16 –Q1

Write a program to sort a matrix using insertion sort and check if the resultant matrix is upper triangular or not

object UpperTrang {

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

{

var A=Array.ofDim[Int](100,100)

println("Enter no. of rows :: ")

var m= scala.io.StdIn.readInt()

println("\nEnter no. of cols :: ")

var n = scala.io.StdIn.readInt()

printf("\nEnter values to the matrix :: \n")

for (i <- 0 to m-1)

{

for (j <- 0 to n-1)

{

A(i)(j)=scala.io.StdIn.readInt()

println(A(i)(j)+" ");

}

}

println("\nThe given matrix is ::");

for (i <- 0 to m-1)

{

for (j <- 0 to n-1)

{

print(A(i)(j)+"\t");

}

print("\n\n");

}

var isUpper = 1;

for(row<-0 to m-1)

{

for(col<-0 to n-1)

{

if(col<row && A(row)(col)!=0)

{

isUpper = 0;

}

}

}

if(isUpper==1)

{

println("\nThis is a Upper triangular matrix")

for(row<-0 to m-1)

{

for(col<-0 to n-1)

{

if(A(row)(col)!= 0)

{

print(A(row)(col)+"\t");

}

else

{

print("\t");

}

}

print("\n\n");

}

}

else

{

print("\nThis is Not a Upper triangular matrix.");

}

}

}

SLIP 17 – Q1

Create abstract class Shape with abstract functions volume() and

display(). Extend two classes Cube and Cylinder from it. Create

objects of Cube and Cylinder, Calculate volume of each and displayit.

(create abstract class :5M extend classes : 5M functions: 5M each

display: 10M)

import scala.annotation.switch

abstract class shape()

{

    def volume()

    def display()

}

class cube(side:Int) extends shape()

{

    var s:Int=side

    var v:Int=0

    def volume()

    {

        v=s\*s\*s

    }

    def display()

    {

        println("volume is:"+v)

    }

}

class cylinder(radius:Int,height:Int) extends shape()

{

    var r:Int=radius

    var h:Int=height

    var v:Float=0.0f

    var pi=3.14f

    def volume()

    {

         v=pi\*(r\*r\*h)

    }

    def display()

    {

        println("volume of cylinder is:"+v)

    }

}

object slip10

{

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

    {

        println("Enter your choice:1)cube 2)cylinder 3)exit:")

        var ch:Int=scala.io.StdIn.readInt()

        ch match

        {

            case 1 => println("Enter side:")

                    var s:Int=scala.io.StdIn.readInt()

                    var c=new cube(s)

                    c.volume()

                    c.display()

            case 2 => println("Enter raadius:")

                      var r:Int=scala.io.StdIn.readInt()

                      println("Enter height:")

                      var h:Int=scala.io.StdIn.readInt()

                      var cy=new cylinder(r,h)

                        cy.volume()

                        cy.display()

            case  3 => System.exit(1)

        }

    }

}

SLIP 18 – Q1

Create array of strings and read a new string from user. Display all the srings from the array that contain the new string. (accept array and string: 10M check string logic : 10M display 10M)

/ Scala program to adding elements in an array

// of the string as name.

object GFG

{

// Main method

def main(args: Array[String])

{

var name = new Array[String](4)

// Adding element in an array

name(0)="gfg"

name(1)="geeks"

name(2)="GeeksQuize"

name(3)="geeksforgeeks"

println("After adding array elements : ")

for ( m1 <-name )

{

println(m1 )

}

}

}

SLIP 19-Q1

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. (random numbers list: 10M create list with function: 10M display with ascending order : 10M)

object GFG

{

// Main method

def main(args: Array[String])

{

// variable declaration (assigning 5 to a)

var a = 1

// loop execution

while (a<11)

{

val r = new scala.util.Random

println("This is random\_no :" +r.nextInt(30))

println( "This is calculated :"+ 3\*r.nextInt(30)\*r.nextInt(30)+4\*r.nextInt(30)+6)

a=a+1;

}

}

}

SLIP 20 – Q1

Create a MAP for storing the following information about 5 students, where each Student is described with Rollno, Name and marks in two subjects. Display Student information with highest percentage.(HERE U HAVE TO ADD PERCENTAGE AND SUBJECTS )

object Slip1

{

def main(a:Array[String])

{

var map=Map(1->"mayuri",2->"mayuri",3->"Chinmay",4->"mayuri",5->"mayi");

for((k1,v1) <- map)

{

for((k2,v2)<-map)

{

if(v1==v2 && k1!=k2)

{

println(" Roll No: "+ k1+" FirstName:"+v1);

}

}

}

}

}