```
object slip1
{
    def main(args: Array[String])
    {
        var str=""
        var emp= Map("mr.joshi"->"management",
                     "mr.rathi"->"management",
                     "mr.gaikar"->"technical",
                     "mr.kanojiya"->"h.r",
                     "mr.batham"->"finance",
                     "mr.malani"->"marketing")


    println("key         values")
        for ((k,v) <- emp)
        println(k,v)

    println("person having same department as mr.joshi")
        for ((k,v) <- emp)
        {
            if(k=="mr.joshi")
            {
                    str=v
            }
            if(v==str)
            {
                println(k,v)
            }
        }
    }
}
```

department.txt

```
>mongo
>show dbs
>use department
>db.createCollection('student')

db.student.insert({name:"Abhi",course:[{coursename:"bcs"},{coursename:"bvoc"}],marks:80,age:21,gender:"male",city:"pune"})


db.student.insert({name:"mukesh",course:[{coursename:"bcs"},
{coursename:"bvoc"}],marks:60,age:22,gender:"male",city:"pune"})

db.student.insert({name:"manisha",course:[{coursename:"mcs"},
{coursename:"bvoc"}],marks:90,age:22,gender:"female",city:"mumbai"})

db.student.insert({name:"manasi",course:[{coursename:"mcs"},
{coursename:"bvoc"}],marks:92,age:22,gender:"female",city:"latur"})


db.student.insert({name:"apurva",course:[{coursename:"mcs"},
{coursename:"bvoc"}],marks:37,age:22,gender:"female",city:"sasvad"})


db.student.insert({name:"arati",course:[{coursename:"mcs"},
{coursename:"bvoc"}],marks:32,age:22,gender:"female",city:"bekarai"})


a) >
db.student.count({marks:{$gt:80}})

b) >
 db.student.find({marks:{$lt:40}})

c) >
var my=db.student.find({marks:{$gt:70}});while(my.hasNext()){print(tojson(my.next()));}

d)>
db.student.find({gender:"female",$or:[{city:"pune"},{city:"mumbai"},{marks:{$lt:50}}]})
```

Prg2.txt

```
object Prg2
{
	def main(args:Array[String])
	{
		var occ:Int=0
		println("Enter String:");
		var str1=Console.readLine();

		println("Enter String:");
		var str2:Char =Console.readChar;

		for(i<-0 until str1.length)
		{
			if(str1(i)==str2)
			{
				if(str1(i).isLower)
				{
					var c=Character.toUpperCase(str1(i))
				}
					else
					{
						str1(i).toUpper
					}
				occ+=1
			}
		}
		println("occurence: "+occ)

	}
}
```

slip2_1.txt

```
object slip2_1
{
    def main(args: Array[String])
    {
        println("enter five random numer")
        for(i<-0 until 5)
        {
            var a =Console.readInt
            println("BINARY OF :"+a)
            binary(a)
            println("octal OF :"+a)
            octal(a)
        }
        def binary(n:Int)
        {
            var binary = Integer.toBinaryString(n);
            println(binary)
        }
        def octal(n:Int)
        {
            var oct = Integer.toOctalString(n);
            println(oct)
        }
    }
}
```

slip3.txt

```
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 slip3
{
        def main(args:Array[String])
        {
                        var c1=new SalesOrder(1,"Two Laptops",200,"XYZ",233221);
                        var s1=new PurchaseOrder(2,"Three Computers",101,"ABC",211231);
                        println("Purchase Order");
                        println("----------------------------------------");
                        c1.display();
                        println("Sales Orders");
                        println("----------------------------------------");
                        s1.display();
        }
}
```

slip2_2.txt

```
object slip2_2
{
    def main(args: Array[String])
    {
        var cnt=0;
        var sum=0.00;
        var j=2;
        println("enter N1 and N2 ")
        var n1: Int= Console.readInt
        var n2: Int= Console.readInt
        for(i<-n1 until n2)
        {
                if(i%j==0)
                {
                  sum+=i
                    cnt+=1
                }
        }
        println("average of prime Number between $n1 & $n2: "+(sum/cnt) )
    }
}
```

3 books.txt

```
>mongo
>show dbs
>use department
>db.createCollection('book')

db.book.insert({BName:"shyamchi aai",cost:700,author:"sane guruji",published:2007})

db.book.insert({BName:"TwoSaints",
cost:1700,author:"raguramkrishna",published:2017})

db.book.insert({BName:"ramkrushna paramhans", cost:800,author:"raguramkrishna",published:2017})

db.book.insert({BName:"DMS",cost:300, author:"raguramkrishna" ,published:2005})

db.createCollection('publisher')

db.publisher.insert({pname:"O Reilly",language:"English",books:[{BName:"ramkrushna paramhans"},{BName:"Two
Saints"}],city:"mumbai"})

db.publisher.insert({pname:"vision",language:"English",books:[{BName:"DMS"}],city:"pune"})

db.publisher.insert({pname:"O Reilly",language:"marathi",books:[{BName:"shyamchi aai"}],city:"mumbai"})


a)>
db.publisher.find({city:"mumbai"})

b)>
db.book.find({cost:{$lt:1000}})

c) >
db.book.find({author:"raguramkrishna",published:2017})

d)>
 db.publisher.find({pname:"O Reilly",$or:[{language:"English"},{language:"marathi"}]})
```

4 hosp.txt

```
>mongo
>show dbs
>use hosp
>db.createCollection('Hospital')
db.Hospital.insert({Hno:1,Hname:"AAA",Specialization:["Pediatric","Gynaec","Orthopaedic"],People:[{Pname:"PQR",Rating:4},
{Pname:"SDE",Rating:5}],Doctor:[{"Dname" : "WWW","Visit" : "Sunday"}]})

db.Hospital.insert({Hno:2,Hname:"BBB",Specialization:["Gynac","Orthopaedic"],People:[{Pname:"POP",Rating:2},{Pname:
"SDE",Rating:3}],Doctor:[{"Dname":"XXX",Visit:"Monday"}]})

db.Hospital.insert({Hno:3,Hname:"CCC",Specialization:["Gynac","Orthopaedic","Pediatric"],People:[{Pname:"KLO",Rating:3},
{Pname:"LPO",Rating:3}],Doctor:[{"Dname" : "XXX","Visit":
"Tuesday"}]})

a) >
db.Hospital.find({Specialization:"Pediatric"})

b)>
db.Hospital.find({Hname:"CCC","Doctor.Visit":"Tuesday"})

c)>
db.Hospital.find({Specialization:{$not:{$size:1}},"Doctor.Dname":"XXX"})

d) >
 db.Hospital.find({"People.Rating":{ $gt: 3 },Hname:"AAA"})
```

5 blog.txt

```
>mongo
>show dbs
>use blog
>db.createCollection('post')

db.post.insert({title:"online",url:"www.abc.com",tag:["food","travel"],pname:"mukesh",pdate:new Date("2019-03-12"),
like:89,user:[{name:"abhi",comment:"good",message:"do best", cdate:new Date("2020-03-12"),like:1}]})

db.post.insert({title:"wetpet",url:"www.wetpet.com",tag:["food","travel",],pname:"Amit",pdate:new Date("2018-03-
12"),like:82,user:[{name:"abhi",comment:"good",message:"do best",time:"4pm",like:1},
{name:"mukesh",comment:"best",message:"success", cdate:new Date("2008-11-12"),like:2}]})

db.post.insert({title:"wetpet",url:"www.wetpet.com",tag:["food","travel","magic"],pname:"abhijeet",pdate:new Date("2017-03-
12"),like:182,user:[{name:"sagar",comment:"like",message: "do best",time:"4pm",like:1},
{name:"mukesh",comment:"best",message:"success", cdate:new Date("2019-03-12"),like:2}]})

db.post.insert({title:"nonveg",url:"www.non.com",tag:["food","travel","chiken"],pname:"Amit",pdate:new Date("2019-07-12"),
like:82,user:[{name:"manisha",comment:"good",message:"do best",time:"4pm",like:0},
{name:"manasi",comment:"best",message:"success", cdate:new Date("2018-03-12"),like:0}]})



a) >
db.post.find({tag:"food"})

b) >
db.post.find({pname:"Amit"})

c) >
db.post.find({tag:"travel",pdate:{"$lte":new Date("2018-03-11")}, "user.name":"sagar","user.comment":"like"})
d) >
db.post.find({$or:[{"user.cdate":{$lte:new Date("2019-08-07")}},{"user.like":0}]})
```

```
object slip5
{
    def main(args:Array[String])
    {
        val a=Set("hello","where","will","when","your")
        val b=Set("hello","fine","okay","are","you")
        println(a)
        println(b)
       var c=a.intersect(b)
       println("common elements in given set : "+c)
       var lm=a.filterNot(b.contains(_))
       var km=b.filterNot(a.contains(_))
       println("after merging both set:"+(lm++km))

    }
}
```

```
import scala.collection.mutable.Set
object Prg6
{
        def main(args:Array[String])
        {
                        var  s1=Set(1,2,3,4,5,6);
                        var s2=Set(4,5,6,7,8);

                        println("set1"+s1)
                        println("set1"+s2)
                        var s3=s1++s2
                        println("merged sets S1 & S2")
                        println(s3)
                        println("sum of all integers in the merged set:"+(s3.sum))
                        println("minimum from set:"+s3.min)
                        println("maximum from set:"+s3.max)
        }
}
```

6 mongo tours.txt

```
>mongo
>show dbs
>use tours

db.createCollection('turism')
>db.turisum.insert({name:"veena word",rate:9,package:[{pname:"shillong",cost:10000},{pname:"gujart",cost:7000},
{pname:"karnataka",cost:6000}]})

>db.turisum.insert({name:"rohit",rate:7,package:[{pname:"shillong",cost:10000},{pname:"rujan",cost:7000}]})

>db.createCollection('tour')

>db.tour.insert({sourc:"john",destination:"shillong",toerisumName:"veenaword",tourisumrate:8000,expense:20000,year:2018,cus
tomer:[{cname:"mukesh",city:"pune"},{cname:"abhijeetsangita",city:"baramati"},{cname:"manisha",city:"15no"},
{cname:"manasi",city:"latur"}]})

>db.tour.insert({sourc:"john",destination:"karnataka",toerisumName:"veenaword",tourisumrate:80090,expense:20900,year:2017,c
ustomer:[{cname:"mukesh",city:"pune"},{cname:"abhijeetsangita",city:"baramati"},{cname:"manisha",city:"15no"},
{cname:"manasi",city:"latur"}]})
>
db.tour.insert({sourc:"john",destination:"rajasthan",toerisumMukesh A. Patil Page
11Name:"rohit",tourisumrate:6000,expense:30400,year:2019,customer:[{cname:"mukesh",city:"pune"},
{cname:"abhijeetsangita",city:"baramati"},{cname:"manisha",city:"15no"},{cname:"manasi",city:"latur"}]})
>
db.tour.insert({sourc:"john",destination:"taj",toerisumName:"rohit",tourisumrate:60090,expense:10400,year:2016,customer:
[{cname:"mukesh",city:"pune"},{cname:"abhijeetsangita",city:"baramati"},{cname:"manisha",city:"15no"},
{cname:"manasi",city:"latur"}]})
4)
a) >db.turisum.find({name:"veena word"}).pretty()
b) >db.turisum.find({}).sort({"rate":-1}).limit(1)
c) >
db.tour.aggregate([{"$sort":{"year":1}},{"$limit":3},{$group:{
_id:null,"count":{"$sum":"$expense"}}}])
d) > db.tour.find({destination:"shillong"})
```

slip7.txt

```
object slip7
{
	def main(args:Array[String])
	{
		println("Enter rows and colums")
		var r:Int=Console.readInt
		var c:Int=Console.readInt
		val arr1=Array.ofDim[Int](r,c);//1st array
		val arr2=Array.ofDim[Int](r,c);//2nd array
		var	rarry=Array.ofDim[Int](r,c)//resultant Array
		  var isupper=1;

		println("Enter Matrix1");
		for(i<-0 until r)
		{
			for(j<-0 until c)
			{
				arr1(i)(j)=Console.readInt();//read Array1 element
			}
		}
		println("Enter Matrix2");
		for(i<-0 until r)
		{
			for(j<-0 until c)
			{
				arr2(i)(j)=Console.readInt();//read Array2 element
			}
		}
		println("MATRIX -1");

		for(i<-0 until r)
		{
			for(j<-0 until c)
			{
				print(arr1(i)(j)+" ");//print Array Element
			}
			println();
		}
		println("MATRIX -2");

		for(i<-0 until r)
		{
			for(j<-0 until c)
			{
				print(arr2(i)(j)+" ");//print Array Element
			}
			println();
		}
		for(i<-0 until r)
		{
			for(j<-0 until c)
			{
				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 until r)
		{
			for(j<-0 until c)
			{
				print(rarry(i)(j)+" ");//print Array Element
			}
			println();
		}
for(i<-0 until r)
		{
			for(j<-0 until c)
			{
				if((i>j)&&(rarry(i)(j)!=0))
					isupper=0;
			}
		}
		if(isupper==1)
			println("Matrix is upper triangular");
		else
			println("Matrix is not upper triangular")


		}}
```

T mongo scientist.txt

```
>mongo
>show dbs
>use scientist

> db.createCollection('scien')

db.scien.insert({fname:"abhi",lname:"nalave",BOD:new Date("1972-04-18"),DOD:"stillalive",field:["tcs","java","sql"],award:
[{name:"codemaster",year:1976},{name:"robot",year:1998},{name:"puzzletalent",year:1995}]})

db.scien.insert({fname:"mukesh",lname:"navse",BOD:new Date("1952-04-18"),DOD:"stillalive",field:
["tcs","java","c","sql"],award:[{name:"turingmachine",year:1976},{name:"robotic",year:1998},{name:"code
talent",year:1995}]})

db.scien.insert({fname:"manisha",lname:"hipparkar",BOD:new Date("1942-04-18"),DOD:new Date("2009-08-06"),field:
["tcs","java"],award:[{name:"topper",year:1976},{name:"puraskar",year:1998},{name:"puzzletalent",year:1995}]})

a) > db.scien.find({ lname: { $regex: /n/ } })
b) > db.scien.find({BOD:{"$gt":new Date("1950-03-11")},DOD:"still alive"})
c)
>db.scien.aggregate([{$group:{_id:{year:"$award.year",Name:"$award.name"}}}])
d) > db.scien.find({"award.name":"turingmachine","award.year":{$lt:1980},field:{$size:4}})
```

Slip 8 create Array of String from user accept new string

```
object Test
{
  def main(args: Array[String])
  {
    var str = Array("Hello", "Good Morning",
                    "Good Afternoon");
    println("Enter String");
    var str1 = scala.io.StdIn.readLine();
    for(i <- 0 to str.length-1)
    {
      if(str(i) == str1)
        println("string is present in
                 array at "+i+"and
                 "+str(i))
      else
        println("not present at "+i+"and
    }
  }
}
```

Seq in Scala Collection →
It maintain insertion order of elet.
It finds the occurance of elet & return a list

Vector → general-purpose, immutable data structure. It provides random access to elets.

8 inventory.txt

```
>mongo
>show dbs
>use inventory
>db.createCollection('item')
>db.item.insert({itemName:"planner",tag:["wash","food","vehicle"],status:"A",height:5,width:9,instack:15,warehouse:
[{location:"pune",quntity:36},{location:"mumbai",quntity:67}]})
>db.item.insert({itemName:"toycar",tag:["food","vehicle"],status:"D",height:5,width:9,instack:15,warehouse:
[{location:"pune",quntity:36},{location:"mumbai",quntity:67}]})
>db.item.insert({itemName:"roboticcar",tag:["food","vehicle"],status:"A",height:9,width:9,instack:5,warehouse:
[{location:"pune",quntity:26},{location:"mumbai",quntity:17}]})
>db.item.insert({itemName:"bag",tag:
["food","vehicle","school","travel"],status:"c",height:19,width:39,instack:75,warehouse:[{location:"surat",quntity:26},
{location:"lanavala",quntity:17}]})


4)
a) > db.item.find({status:"D","warehouse.quntity":{$gt:30}})
b) > db.item.find({"tag":{$size:3}})
c) >
db.item.find({$or:[{status:"A"},{"warehouse.quntity":{$lt:30},height:{$gt:10}}]})
d) > db.item.find({itemName:"planner",instack:{$lt:20}})
```

Prg9.txt

```
object Prg9
{
	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)=Console.readInt();
			}
		}

		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");
	}
}
```

9 transzction.txt

```
>use trans
>
db.transaction.insert({itemName:"toy",customerName:"john",paymentmode:"debitcard",payment:8000})

>
db.transaction.insert({itemName:"car",customerName:"john",paymentmode:"creditcard",payment:4000})

>
db.transaction.insert({itemName:"bag",customerName:"mukesh",paymentmode:"cash",payment:5000})

>
db.transaction.insert({itemName:"airlineticket",customerName:"rohit",paymentmode:"cash",payment:50090})

>
db.transaction.insert({itemName:"mango",customerName:"abhijeet",paymentmode:"creditcard",payment:8000})


db.transaction.insert({itemName:"bus",customerName:"manasi",paymentmode:"debitcard",payment:7000})

4)
a) > db.transaction.find({customerName:"john"})
b) > db.transaction.find({paymentmode:"debitcard"})
c) >
db.transaction.aggregate([{$match:{"paymentmode":"creditcard"}},{$group:{_id:null,"count":{"$sum":"$payment"}}}])
d) >
db.transaction.aggregate([{$group:{_id:"$paymentmode","count":{"$sum":"$payment"}}}])
```

10 mobile.txt

```
>mongo
>show dbs

> use mobile

> db.createCollection('custome')
>db.custome.insert({cname:"mukesh",modelname:"samsungj6",amount:20000})

>
db.custome.insert({cname:"abhijeet",modelname:"samsungj6",amount:20060})

> db.custome.insert({cname:"manasi",modelname:"iphone7+",amount:30060})

> db.custome.insert({cname:"manisha",modelname:"iphone7+",amount:30070})

> db.custome.insert({cname:"dipak",modelname:"iphone7+",amount:30800})

>db.createCollection('shopping')
>
db.shopping.insert({brandname:"samsung",rate:6,model:[{mname:"s40",ram:"3GB",rom:"32GB",rate:4},
{mname:"j6",ram:"4GB",rom:"32GB",rate:7},{mname:"j7",ram:"6GB",rom:"64GB",rate:6}]})

>
db.shopping.insert({brandname:"vivo",rate:8,model:[{mname:"Y55",ram:"3GB",rom:"32GB",rate:6},
{mname:"Ys5",ram:"4GB",rom:"32GB",rate:4},{mname:"YYY",ram:"6GB",rom:"64GB",rate:6}]})

4)
a) >
db.shopping.find({"model.ram":"3GB","model.rom":"32GB"})
b) > db.custome.find({modelname:"samsungj6"})
c) > db.shopping.aggregate([{"$sort":{"rate":-1}},{"$limit":1},{$group:{_id:"$brandname"}}])
d) > db.custome.find().sort( { "cname": 1 } )
```

```
object Prg10
{
        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=Console.readLine();

                var str1=new StringBuilder(str);
                println("Enter character:");
                ch=Console.readChar();

                str1.deleteCharAt(str1.indexOf(ch.toString()));
                var str3=str1.deleteCharAt(str1.lastIndexOf(ch.toString())).toString;

                var str4=str3.map(reverseString)
                println(str4);

        }
}
```

Skill

Find occu of 'x' in string
Reverse the case of each occurrence in the string

Date
Page

object Test

```
{ def main(args: Array[String])
  {
    println("Enter 1 string")
    var str1 = scala.io.StdIn.readLine()
    pr                    2 string
    var str2 = scala.io.StdIn.readLine()
    For (i <- 0 to str1.length)
    { For(j <- 0 to str2.length)
      {
        if(str1(i) == str1(j))
        {
          println("string" + str1(i))
          str3 = str1.map(reversestring(c))
        }
      }
    }
```

Transformed string to new collection which contain reverse string.

using map() method by passing user-defined fn as parameter.

```
def reversestr(str: String): String =
{
  if(str.isLower)
    str.toUpper
  else
    str.toLower
}
```

```
if(str1(i) == str2(j))
{ if(str1(i).isLower)
  var c = Character.
    toUpperCase(str1(i))
}
else
{
  str1(i).toUpper
}
}
}
```

```
var str3
if(str1(i).isUpper)
  str3 = str1(i).toLower
else
  str3 = str1(i).toUpper
println(" str3 ) // Reverse string.
```

neo_11.txt

```
reate (p1:person{name:"deepak"}),
       (p2:person{name:"saurabh"}),
       (p3:person{name:"manoj"}),
       (p4:person{name:"usha"}),
       (c1:city{city_name:"pune"}),
       (c2:city{city_name:"mumbai"}),
       (c3:city{city_name:"kholapur"}),
       (pro1:project{pro_name:"Finance"}),
       (pro2:project{pro_name:"inventory"}),
       (pro3:project{pro_name:"sale"})

reate (p2)-[:sibling{relation:"brother"}]->(p1),
       (p4)-[:parent{relation:"mother"}]->(p1),
       (p4)-[:parent{relation:"mother"}]->(p2),
       (p3)-[:parent{relation:"father"}]->(p1),
       (p3)-[:parent{relation:"father"}]->(p2),
       (p1)-[:stays]->(c1),
       (p2)-[:stays]->(c2),
       (p3)-[:stays]->(c3),
       (p4)-[:stays]->(c3),
       (p1)-[:working_on]->(pro1),
       (p2)-[:working_on]->(pro2),
       (p3)-[:working_on]->(pro3) return p1,p2,p3,p4,c1,c2,c3,pro1,pro2,pro3

: match(per:person)-[p:parent]->(per1:person) return per
: match(per:person)-[:working_on]->(pro:project) where pro.pro_name="Finance" return per
: match(p:person)-[:stays]->(c:city) where c.city_name="mumbai" or c.city_name="pune" return p
: match(p:person)-[pp:parent]->(p1:person) where pp.relation="mother"  return p
```

neo_13.txt

```
create (c1:country{name:"india"}),
       (c2:country{name:"china"}),
       (s1:state{name:"maharashta"}),
       (s2:state{name:"gujurat"}),
       (s3:state{name:"bejing"}),
       (s4:state{name:"yon tho"}),
       (p1:product{name:"oil"}),
       (p2:product{name:"sugarcane"})

create (c1)-[:has_state]->(s1),
       (c1)-[:has_state]->(s2),
       (c2)-[:has_state]->(s3),
       (c2)-[:has_state]->(s4),
       (c1)-[:produce{percentage:"75"}]->(p2),
       (c1)-[:produce{percentage:"50"}]->(p1),
       (c2)-[:produce{percentage:"60"}]->(p2),
       (c2)-[:produce{percentage:"55"}]->(p1),
       (c1)-[:export]->(p1),
       (c1)-[:export]->(p2),
       (c2)-[:import]->(p1),
       (c2)-[:import]->(p2) return c1,c2,s1,s2,s3,s4,p1,p2

A: match(c:country)-[:export]->(p:product) where p.name="oil" return c
B: match(p:product)<-[:produce]-(c:country)-[:has_state]->(s:state) where s.name="maharashta" return p
C: match(c:country)-[p1:produce]->(p:product) where p1.percentage > "70" return c
D: match(c:country)-[:export]->(p:product) return c
```

```
object Prg13
{
        def reverse(ch:Char):Char=
        {
                if(ch.isLower)
                        ch.toUpper;
                else
                        ch.toLower;
        }
        def main(args:Array[String])
        {
                var str=" ";
                println("Enter String:");
                str=Console.readLine();
                var str2=str.map(reverse);
                println(str2);
        }
}
```

Slip 16 user defined fun to reverse each char call using map

```
object Test
{
def reverse(ch: Char): Char =
{ if(ch.isLower)
    ch.toUpper
  else ch.toLower
}
def main(args: Array[String])
{
  println("Enter string")
  var str = scala.io.StdIn.readLine()
  var str2 = str.map(reverse)
  println(str2)          // transformed string to map.
}
}
```

Here, we are passing a user-defined fun reverse as a parameter to map() method Using this a new collection str2 is created which contain reverse of string.

neo_16.txt

```
create(h1:hotel{name:"raj hotel",address:"camp area"}),
        (h2:hotel{name:"maharaja hotel",address:"koregaon park"}),
        (h3:hotel{name:"kamal hotel",address:"koregaon park"}),
        (f1:facility{type:"lodging"}),
        (f2:facility{type:"restaurant",type1:"lodging"}),
        (p1:person{name:"deepak"}),
        (p2:person{name:"akash"}),
        (p3:person{name:"shurti"}),
        (p4:person{name:"bhavya"})

create (h1)-[:has_facilities]->(f1),
        (f2)<-[:has_facilities]-(h2)-[:has_facilities]->(f1),
        (h1)<-[:visted]-(p1)-[:rated{rating:"4.5"}]->(h1),
        (h1)<-[:visted]-(p2)-[:rated{rating:"4"}]->(h1),
        (h2)<-[:visted]-(p3)-[:rated{rating:"3"}]->(h2),
        (h2)<-[:visted]-(p4)-[:rated{rating:"3.5"}]->(h2),
        (h2)<-[:visted]-(p1)-[:rated{rating:"4"}]->(h2),
        (p1)-[:recommendend]->(h1)-[:to]->(p2),
        (p2)-[:recommendend]->(h2)-[:to]->(p3),
        (p3)-[:recommendend]->(h2)-[:to]->(p1),
        (p4)-[:recommendend]->(h3)-[:to]->(p2) return h1,h2,h3,f1,f2,p1,p2,p3,p4

A: match(h:hotel) where h.address="camp area" return h
b: match(h:hotel)-[:has_facilities]->(f:facility) where f.type1="lodging" and f.type="restaurant" return h
c: match(p:person)-[r:rated]->(h:hotel) where r.rating >="4" return h
D: match(p:person)-[r:recommendend]->(h:hotel) where h.address="koregaon park" return h,count(r) as cnt
```

```
abstract class Shape
{
        def volume():Double;
        def display();
}
class Cylinder(var r:Int,var h:Int) extends Shape
{
        def volume():Double=
        {
                return 3.14*r*r*h;
        }
        def display()
        {
                println("Volume Cylinder :"+volume());
        }
}
class Cube(var s:Int) extends Shape
{
        def volume():Double=
        {
                return s*s*s;
        }
        def display()
        {
                println("Volume of cube:"+volume());
        }
}
object Prg17
{
        def main(args:Array[String])
        {
                val cyl=new Cylinder(1,1);
                cyl.display();

                val cub=new Cube(3);
                cub.display();
        }
}
```

neo_IT.txt

```
create(d1:doctor{name:"deepak verma"}),
    (d2:doctor{name:"rajiv bhatia"}),
    (s1:specialization{name:"gynaec"}),
    (s2:specialization{name:"orthopedic"}),
    (s3:specialization{name:"pediatric"}),
    (h1:hospital{name:"verma hospital"}),
    (h2:hospital{name:"jehangir hospital"}),
    (p1:person{name:"raju sharma"}),
    (p2:person{name:"rajiv shukla"}),
    (p3:person{name:"amitabh bachan"})

create (h1)-[:specialist]->(s1),
        (h2)-[:specialist]->(s1),
        (h2)-[:specialist]->(s2),
        (h1)-[:specialist]->(s3),
        (p1)-[:reviwed{rated:"3.5"}]->(h2),
        (p2)-[:reviwed{rated:"4"}]->(h1),
        (p3)-[:reviwed{rated:"3"}]->(h2),
        (p2)-[:reviwed{rated:"3.5"}]->(h2),
        (d1)-[:visits{day:"monday"}]->(h1),
        (d1)-[:visits{day:"tuesday"}]->(h2),
        (d2)-[:visits{day:"friday"}]->(h1),
        (d2)-[:visits{day:"saturday"}]->(h2) return d1,d2,s1,s2,s3,h1,h2,p1,p2,p3

A: match(h:hospital)-[:specialist]->(s:specialization) where s.name="pediatric" return h
B: match(d:doctor)-[v:visits]->(h:hospital) where v.day="monday" return d
C: match(p:person)-[r:reviwed]->(h:hospital)-[:specialist]->(s:specialization) where s.name="gynaec" return h, count(r)
D: match(p:person)-[r:reviwed]->(h:hospital) where r.rated >="3" and h.name="jehangir hospital" return p
```

slip17.txt

```
class CurrentAccount(var accno: Int,var name: String,var balance: Double,var minbal: Int )
{
        def viewbalance()
        {
            println("CURRENT BALANCE:"+balance);
        }
        def withdraw()
        {
            println("HOW MUCH AMOUNT YOU WANT TO WITHDRAW");
            var rmv : Int = Console.readInt;
            if(rmv > balance)
            {
                println("AMOUNT IS MORE THAN YOUR BALANCE.\n PLS ENTER VALID AMOUNT");
            }
            else
            {
            balance = balance-rmv;
            println("AMOUNT WITHDRAWED :" +rmv);
            println("REMAINING BALANCE: "+balance);
            }
        }
        def deposit()
        {
            println("ENTER AMOUNT TO BE DEPOSITED");
            var amt : Int = Console.readInt;
            balance = balance+amt;
            println("AMOUNT DEPOSITED :" +amt);
            println("CURRENT BALANCE: "+balance);
        }
}

object slip17
{
            def main(args: Array[String])
            {
                var ch: Int = 0;
                println("enter your bank Details");
                println("ENTER ACCOUNT NO:")
                val acc: Int= Console.readInt;
                println("ENTER ACCOUNT HOLDER NAME:")
                var name: String= Console.readLine;
                println("ENTER CURRENT BALANCE");
                var bal: Double = Console.readDouble;
                var min: Int = 2000;
                var obj= new CurrentAccount(acc,name,bal,min);
                do
                {
                    print("1.VIEW BALANCE \n 2.WITHDRAW \n 3.DEPOSIT \n 4.EXIT\n");
                    println("enter your choice");
                    ch= Console.readInt;
                    ch match{
                        case 1=> obj.viewbalance();
                        case 2=> obj.withdraw();
                        case 3=> obj.deposit();
                        case 4=> "";
                    }
                }while(ch!=4)

            }

}
```

clip 18

sum of perfect no bet^n 1 & 100
Display perfect no. ajv (sum of its proper +ve divisors is equal to no itself excluding self division)

```
object Test
{
  def main(args: Array[String])
  {
    var sum=0
    var pSUm:=0
    var perfect=""
    For(i<-0 to 100
    {
      For(j<- 1 to i-1)
      {
        if(i%j==0)
        { sum=sum+j;
        }
      }
      if(sum==i)
      {
        psum=psum+i
        // sum of perfect no
        perfect=perfect+" "+i
      }
      sum=0;
    }
    println(" Perfect no "+perfect)
    println(" Sum of Perfect no "+psum)
  }
}
```

6 = 2+3+1
28 = 1+2+4+7+14
496 =
8128 =
33550336 =

6 = 2+3+1
10 = 1+2+5
8 = 4+2+1
12 = 4+2+3+1+6
16 = 4+2+1+9

clip 19

```
object
{
  de
}
```

② list
objec
{ def
}

Prg18.txt

```
object Prg18
{
	def main(args:Array[String])
	{
		val str:Array[String]=Array("Hello Good Morning","Hello Good Night","Hello Good Afternoon");
		var str1=" ";

		println("Enter string:");
		str1=Console.readLine();
		for(i<-0 until str.length)
		{
			if(str(i).contains(str1))
			{
				println(str)
			}
		}

	}
}
```

```
object Prg7
{
        def main(args:Array[String])
        {
                        println("A:(List style, java style, fill, range, tabulate methods)")
                var l1:List[Int]=1::2::3::Nil;//Lisp Method
                println(l1);

                var l2:List[Int]=List(1,2,3);//Java Method
                println(l2);

                var l3:List[String]=List.fill(3)("HELLO");//Fill
                println(l3);

                var l4:List[Int]=List.range(1,5);//range
                println(l4);

                var l5:List[Int]=List.tabulate(5)(n=>n*n);//tabulate
                println(l5);

                println("B:list of 50 member using function 2n+3")
                var l6:List[Int]=List.tabulate(50)(n=>2*n+3);
                println(l6)

                var l7:List[Int]=l6.filter(n=>n%5!=0)
                println("elements exculding multiple of 5")
                println(l7)
        }
}
```

neo_19.txt

```
create(v1:vehicle_type{name:"heavy"}),
        (v2:vehicle_type{name:"light"}),
        (vv1:vehicle{name:"royal enfield"}),
        (vv2:vehicle{name:"splendor"}),
        (vv3:vehicle{name:"jeep"}),
        (c1:customer{name:"deepak"}),
        (c2:customer{name:"akash"}),
        (c3:customer{name:"pranav"}),
        (c4:customer{name:"rishi"})

create(v1)-[:has]->(vv1),
        (v1)-[:has]->(vv3),
        (v2)-[:has]->(vv2),
        (c1)-[:bought]->(v1),
        (c2)-[:bought]->(v1),
        (c3)-[:bought]->(v1),
        (c4)-[:bought]->(v2),
        (c2)-[:bought]->(v2),
        (c1)-[:recommended]->(v1),
        (c2)-[:recommended]->(v1),
        (c3)-[:recommended]->(v1),
        (c2)-[:recommended]->(v2),
        (c4)-[:recommended]->(v2) return v1,v2,vv1,vv2,vv3,c1,c2,c3,c4

A: match(v:vehicle_type)-[:has]->(v1:vehicle) where v.name="heavy" return v1
B: match(c:customer)-[:bought]->(v:vehicle_type) where v.name="light" return c
C: match(c:customer)-[b:bought]->(v:vehicle_type) return c, count(b) as cnt, count(v) as cnt1
D: match(c:customer)-[r:recommended]->(v:vehicle_type) return v, count(r) as cnt
```

Prg20.txt

```
class Student(var rno:Int,var sname:String,var sub1:Int,var sub2:Int)
{
	var ptage:Float=(sub1+sub2/200)*100;
	def display()
	{
		println("Roll No:"+rno);
		println("Name:"+sname);
		println("Percentage:"+ptage);
	}
}
object Prg20
{
	def main(args:Array[String])
	{
		val s1=new Student(1,"Ajinkya Bhosale",80,70);
		val s2=new Student(2,"Sahil Bhosale",75,85);
		val s3=new Student(3,"Rushikesh Bacchhnav",77,87);
		val s4=new Student(4,"Subodh Shelke",89,99);
		val s5=new Student(5,"Siddhant Bhadane",84,87);

		val m1:Map[Int,Student]=Map(1->s1,2->s2,3->s3,4->s4,5->s5);

		var max=m1(1).ptage;
		for((k,v)<-m1)
		{
			if(m1(k).ptage>max)
				max=m1(k).ptage;
		}
		for((k,v)<-m1)
		{
			if(m1(k).ptage==max)
				m1(k).display()
		}
	}
}
```