Control Structures :

1) Write a program to calculate average of all numbers between n1 and n2(eg.100 to 300 Read values of n1 and n2 from user)

Ans:

```
import scala.io.StdIn.readInt

object Average_Prime {
   def main(args: Array[String]) {
     var counter: Int=0;

     var  avg =0;
     var sum: Int = 0;
     println("enter first number :");
     val n1= readInt();
     println("enter second number :");
     val n2= readInt();
     var n = n2-n1;
     for(counter <- n1 to n2) {
       sum=sum+counter;
     }
     avg= sum/n;
     println("Average = "+ avg);
   }
 }
```

Output:

```
Enter the Num1: 1
Enter the Num2: 50
Average is: 21.866666666666667
```

2) Write a program to calculate factorial of a number.

Ans:

```
object Factorial {
  def main(args: Array[String]) {
    var f: Int=1;
    println("enter number :");
    val n= scala.io.StdIn.readInt();
    for(i <- 1 to n) {
      f=f * i;
    }
    println(" Factorial of  " + n + " is  = "+ f);
  }

}
```

Output:

Enter number : 4

The factorial of 4 is 24

3) Write a program to read five random numbers and check that random numbers are perfect number or not.

Ans:

```
object  Perfect
{
def main(args : Array[String])
{
var r=scala.util.Random;
var sum =0;
for(i <- 1 to 5)
{
var n = r.nextInt(50)
```

```
for(j <- 1 to n-1)
{
if(n % j == 0)
{
sum += j
}
}
if(sum == n)
println(n + " is a perfect number")
else
println(n + " is not a perfect number")
}
}
}
```

Output:
26 is not a perfect number
47 is not a perfect number
22 is not a perfect number
45 is not a perfect number
1 is not a perfect number

4) Write a program to find second maximum number of four given numbers.

Ans:

```
object SecondMaximum {
def main(args: Array[String]) =
{
var first = 0
var second = 0
for (i <- 1 to 4)
{
print("Enter numbers: ")
var n1 = scala.io.StdIn.readInt()
if (n1 > first) {
second = first
first = n1
}
```

```
else if (n1 > second && n1 < first) {
second = n1
}
}
println("\nThe Second Largest Number in this Array is : " +second)
}
}
```

Output:

```
Enter numbers: 4
Enter numbers: 5
Enter numbers: 8
Enter numbers: 2

The Second Largest Number in this Array is : 5
```

5) Write a program to calculate sum of prime numbers between 1 to 100.

Ans:

```
object PrimeNoSum {
def main(args: Array[String]) =
{
var primesum=0
for(i <- 2 to 100)
{
var sum=0
for(j<-2 to i)
if(i%j==0)
sum=sum+j
if(sum==i)
{
primesum=primesum+i
println("No. is prime no."+i)
}
}
println("sum of prime numbers are "+primesum)
}
```

}

Output:
No. is prime no.2
No. is prime no.3
No. is prime no.5
No. is prime no.7
No. is prime no.11
No. is prime no.13
No. is prime no.17
No. is prime no.19
No. is prime no.23
No. is prime no.29
No. is prime no.31
No. is prime no.37
No. is prime no.41
No. is prime no.43
No. is prime no.47
No. is prime no.53
No. is prime no.59
No. is prime no.61
No. is prime no.67
No. is prime no.71
No. is prime no.73
No. is prime no.79
No. is prime no.83
No. is prime no.89
No. is prime no.97
sum of prime numbers are 1060

6) Write a program to read an integer from user and convert it to binary and octal using user defined functions.

Ans:

object NumberConv {

```
def binaryCon(n : Int) =
{
var i=0
var num=n
var A=new Array[Int](10)
while(num>0)
{
A(i)=num%2
i=i+1
num=num/2
}
println("\nbinary equivalent " )
for(j<-i-1 to 0 by -1)
print(A(j))
}
def octalCon(n:Int)=
{
var i=0
var num=n
var A=new Array[Int](10)
while(num>0)
{
A(i)=num%8
i=i+1
num=num/8
}
println("\nOctal equivalent " )
for(j<-i-1 to 0 by -1)
print(A(j))
}
def main(args: Array[String]) =
{
println("enter no.")
var n=scala.io.StdIn.readInt()
octalCon(n)
binaryCon(n)
}
}
```

Output:
enter no.
15

Octal equivalent
17
binary equivalent
1111

Arrays :

1) Write a program to find maximum and minimum of an array

Ans:

```
object MaxMinOfArray
{
def main(args: Array[String]) ={
print("Enter the size of array: ")
var size = scala.io.StdIn.readInt()
var arr = new Array[Int](size)
print("Enter Elements of Array: \n")
for(i <- 0 to size-1)
{
arr(i) = scala.io.StdIn.readInt()
}
println("Max element is: "+ maximum(size, arr))
println("Min element is: "+ minimum(size, arr))
}
def maximum(size: Int,Arr:Array[Int]): Int= {
var max = 0
for(i<- 0 to size -1 ) {
if(Arr(i)>max) {
max = Arr(i)
}
}
return max
}
def minimum(size: Int,Arr:Array[Int]): Int= {
var min = Arr(0)
for(i<- 1 to size -1 ) {
if(Arr(i)<min) {
min = Arr(i)
}
}
return min
}
}
```

Output:
Enter the size of array: 5
Enter Elements of Array:
4
7
9
2
5
Max element is: 9
Min element is: 2

2) Write a program to calculate transpose of a matrix.

Ans:

```
object TransposeOfMatrix
{
def main(args: Array[String]) ={
var Matrix1 = Array.ofDim[Int](10, 10)
print("Enter no. of rows: ")
val rows = scala.io.StdIn.readInt()
print("Enter No. of columns: ")
val cols = scala.io.StdIn.readInt()
print("Enter values matrix: \n")
getMatrix(rows, cols, Matrix1)
println("The given first matrix is: ")
display(rows, cols, Matrix1)
transpose(rows, cols, Matrix1)
}
def getMatrix(rows: Int, cols: Int, Arr:
Array[Array[Int]]) {
for (i <- 0 to rows - 1) {
for (j <- 0 to cols - 1) {
Arr(i)(j) = scala.io.StdIn.readInt()
}
}
}
def display(rows: Int, cols: Int, Arr:
```

```
Array[Array[Int]]) {
for (i <- 0 to rows - 1) {
for (j <- 0 to cols - 1) {
print(Arr(i)(j) + " ")
}
print("\n")
}
}
def transpose(rows: Int, cols: Int, Arr:
Array[Array[Int]])
{
var Matrix2 = Array.ofDim[Int](10, 10)
for (i <- 0 to rows - 1) {
for (j <- 0 to cols - 1) {
Matrix2(i)(j) = Arr(j)(i)
}
}
println("The transpose of given first matrix is: ")
display(rows, cols, Matrix2)
}
}
```

Output:

```
Enter no. of rows: 2
Enter No. of columns: 2
Enter values matrix:
4
8
3
6
The given first matrix is:
4 8
3 6
The transpose of given first matrix is:
4 3
8 6
```

3) Write a program to calculate determinant of a matrix.

Ans:

```
object Determinant
{
def main(args: Array[String]) =
{
val matrix= Array.ofDim[Int](10,10)
print("Enter Row :")
val r=scala.io.StdIn.readInt()
print("Enter column :")
val c=scala.io.StdIn.readInt()
if(r!=c)
{
println("Rows and columns should be equal , please run once again")
}
else if(r==2 && c==2 || r==3 && c==3)
{
printf("\n\nCalculate the determinant of a " + r + " x " + c + " matrix :\n")
print("..... \n")
printf("Enter the values in the matrix :\n")
for(i<-0 to r-1)
{
for(j<-0 to c-1)
{
matrix(i)(j)=scala.io.StdIn.readInt()
}
}
det(r,c,matrix)
}
else
{
print("This Program calculate only for 2x2 and 3x3 matrix , please run once
again and enter 2 or 3 rows and columns")
}
}
def det(r:Int,c:Int,matrix:Array[Array[Int]])=
{
if(r==2 && c==2)
{
```

```
printf("The matrix is :\n")
for(i<-0 to r-1)
{
for(j<-0 to c-1)
print(matrix(i)(j)+" ");
println( )
}
val determinant= matrix(0)(0)*matrix(1)(1)-matrix(1)(0)*matrix(0)(1)
println("Determinant = "+determinant)
}
else if(r==3 && c==3)
{
printf("The matrix is :\n")
for(i<-0 to r-1)
{
for(j<-0 to c-1)
print(matrix(i)(j)+" ");
println( )
}
var  x=matrix(0)(0)*(matrix(1)(1)*matrix(2)(2)-matrix(1)(2)*matrix(2)(1))
var  y=matrix(0)(1)*(matrix(1)(0)*matrix(2)(2)-matrix(1)(2)*matrix(2)(0))
var z=matrix(0)(2)*(matrix(1)(0)*matrix(2)(1)-matrix(1)(1)*matrix(2)(0))
var Determinant= x - y + z
println("Determinant = "+Determinant)
}
}
}
```

Output:

```
Enter Row :2
Enter column :2


Calculate the determinant of a 2 x 2 matrix :
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Enter the values in the matrix :
1
7
9
4
```

The matrix is :
1 7
9 4
Determinant = -59

4) Write a program to check if the matrix is upper triangular or not.

Ans:

```
object UpperTriangle
{
def display(rows:Int,cols:Int,Arr: Array[Array[Int]])
{
println("The given matrix is:")
for (i <- 0 to rows-1)
{
for (j <- 0 to cols-1)
{
print(Arr(i)(j)+" ")
}
print("\n")
}
}
def isUpperT(rows:Int,cols:Int,Arr: Array[Array[Int]])
{
var isUpper=1
for (i <- 0 to rows-1)
{
for (j <- 0 to cols-1)
{
if(j<i && Arr(i)(j)!=0)
{
isUpper=0
}
}
}
```

```
if(isUpper ==1)
{
println("The matrix entered is upper triangular matrix")
}
else
{
println("The matrix entered is not upper triangular matrix")
}
}
def main(args: Array[String])
{
var A= Array.ofDim[Int](100,100)
println("Enter no. of rows: ")
var rows=scala.io.StdIn.readInt()
println("Enter no. of columns: ")
var cols=scala.io.StdIn.readInt()
if(rows!=cols)
{
println("Rows and columns should be equal!! (Square matrix)")
}
else
{
println("Enter values in the matrix:")
for (i <- 0 to rows-1)
{
for (j <- 0 to cols-1)
{
A(i)(j)=scala.io.StdIn.readInt()
}
}
display(rows,cols,A)
isUpperT(rows,cols,A)
}
}
}
```

Output:

```
Enter no. of rows:
2
Enter no. of columns:
```

```
2
Enter values in the matrix:
1
7
0
3
The given matrix is:
1 7
0 3
The matrix entered is upper triangular matrix
```

5) Write a program to sort the matrix using insertion sort.

Ans:

```
object InsertionSort {
def main(args: Array[String]) {
val array = Array.ofDim[Int](2,2)
array(0)(0)=3
array(0)(1)=2
array(1)(0)=1
array(1)(1)=0
println("Unsorted matrix ")
for(i <- 0 until 2){
for(j <- 0 until 2){
print(array(i)(j)+"\t")
}
println()
}
println("Ascending sorted matrix ")
var temp=0
var a:Int=0
var b:Int=0
for (a <- 0 until 2) {
for (b <- 0 until 2){
if(a==0 && b==0){}
else{
temp = array(a)(b)
```

```
}
var k = a
var l = b
while (k >= 0 && l>=0 && temp < array(a)(b)) {
array(a + 1)(b + 1) = array(a)(b)
k = k-1;
l = l-1;
}
array(a)(b) = temp
print(array(a)(b)+"\t")
}
println()
}
}
}
```

Output:

```
Unsorted matrix
3       2
1       0
Ascending sorted matrix
0       2
1       3
```

6. Write a program for multiplication of two matrices (Validate number of rows and columns before multiplication and give appropriate message)

Ans:

```
object Multiplication
{
def main(args : Array[String])
{
var mat1 = Array.ofDim[Int](10,10)
var mat2 = Array.ofDim[Int](10,10)
var add = Array.ofDim[Int](10,10)
```

```
var sum = 0

println("Enter no. of rows of 1st matrix element")
var r1 = scala.io.StdIn.readInt()
println("Enter no. of columns of 1st matrix element")
var c1 = scala.io.StdIn.readInt()

println("Enter 1st matrix elements")
for(i <- 1 to r1)
for(j <- 1 to c1)
{
mat1(i)(j) = scala.io.StdIn.readInt()
}

println("1st Matrix:")
for(i <- 1 to r1)
{
for(j <- 1 to c1)
{
print(mat1(i)(j)+"\t")
}
print("\n\n")
}

println("Enter no. of rows of 2nd matrix element")
var r2 = scala.io.StdIn.readInt()
println("Enter no. of columns of 2nd matrix element")
var c2 = scala.io.StdIn.readInt()

println("Enter 2nd matrix elements")
for(i <- 1 to r2)
for(j <- 1 to c2)
{
mat2(i)(j) = scala.io.StdIn.readInt()
}

println("2nd Matrix:")
for(i <- 1 to r2)
{
for(j <- 1 to c2)
```

```
{
print(mat2(i)(j)+"\t")
}
print("\n\n")
}

for(i <- 1 to r1)
{
for(j <- 1 to c2)
{
sum = 0
add(i)(j) = 0
sum=mat1(i)(j)+mat2(i)(j)
add(i)(j) = sum
}
}

println("Addition of matrices:")
for(i <- 1 to r1)
{
for(j <- 1 to c2)
{
print(add(i)(j)+"\t")
}
println("\n\n")
}

}
}
```

Output:

```
Enter no. of rows of 1st matrix element
2
Enter no. of columns of 1st matrix element
2
Enter 1st matrix elements
1
2
3
4
```

1st Matrix:
1 2

3 4

Enter no. of rows of 2nd matrix element
2
Enter no. of columns of 2nd matrix element
2
Enter 2nd matrix elements
4
3
6
7
2nd Matrix:
4 3

6 7

Addition of matrices:
5 5

9 11

String :

1) Write program to count uppercase letters in a string and convert it to lowercase and display the new string.

Ans:

```
object StringDemo {
def main(args: Array[String]) {
var count = 0
print("\nEnter the string: ")
val string = scala.io.StdIn.readLine()
for(i<-string){
if(i.isUpper == true) {
count = count+1
}
}
val str = for (c <- string) yield c.toLower
println("\nThis string is '" + string + "'")
println("Number of Uppercase letters in the string are: " +count)
println("\nThe new string is '" + str + "'")
}
}
```

Output:

```
Enter the string: I Love INDIA

This string is 'I Love INDIA'
Number of Uppercase letters in the string are: 7

The new string is 'i love india'
```

2) Write a program to read a character from user and count the number of occurrences of that character.

Ans:

```
object CountDemo {
```

```
def main(args: Array[String]) ={
print("\nEnter the string: ")
val string = scala.io.StdIn.readLine()
print("Enter the character you want to count: ")
val c = scala.io.StdIn.readChar()
val count = string.count(_ == c)
println("\nThis string is '" + string + "'")
println(s"Count of '$c' in the string : " + count)
}
}
object Count_Demo {
def main(args: Array[String]) {
var count = 0
print("\nEnter the string: ")
val string = scala.io.StdIn.readLine()
print("Enter the character you want to count: ")
val c = scala.io.StdIn.readChar()
for(i<-string){
if(i==c)
count = count+1
}
println("\nThis string is '" + string + "'")
println(s"Count of '$c' in the string :  " + count)
}
}
```

Output:

```
Enter the string: Hello World
Enter the character you want to count: l

This string is 'Hellpo World'
Count of 'l' in the string : 3
```

3) Write a program to read two strings. Remove the occurrence of second string in first string.

Ans:

```
object Scala_String {
def test(str1: String, str2: String): String = {
if (str1.length == str2.length)
return str1 + str2;
if (str1.length > str2.length) {
var diff = str1.length - str2.length;
str1.substring(diff, str1.length) + str2;
} else {
var diff = str2.length - str1.length;
str1 + str2.substring(diff, str2.length);
}
}
def main(args: Array[String]): Unit = {
var str1 = "Welcome";
var str2 = "home";
println("The given strings is: " + str1 + " and " + str2);
println("The new string is: " + test(str1, str2));
str1 = "Scala";
str2 = "Python";
println("The given strings is: " + str1 + " and " + str2);
println("The new string is: " + test(str1, str2));
}
}
```

Output:

```
The given strings is: Welcome and home
The new string is: comehome
The given strings is: Scala and Python
The new string is: Scalaython
```

4) Create array of strings and read a string from user. Display all the elements of array containing given string.

Ans:

```
import scala.collection.mutable.ArrayBuffer;
object Ar
{
def main(args: Array[String])=
{
var Animal=ArrayBuffer("Lion", "Tiger", "Elephant", "Deer", "Leopard")
println("1.add, 2.remove, 3.display, 4.length, 5.exit");
println("enter the choice");
var s=scala.io.StdIn.readInt();
while(s<5)
{
s match
{
case 1=>
{ println("At which position would you like to add the element")
var n=scala.io.StdIn.readInt()
var ch=scala.io.StdIn.readLine("enter the element\n")
Animal.insert(n,ch)
}
case 2=>
{ println("enter the position of the element");
var ch1=scala.io.StdIn.readInt()
Animal.remove(ch1)
}
case 3 =>
{ println("The elements of the Array:")
for(i <- 0 to Animal.length-1)
println(Animal(i)+ "")
}
case 4=>
{  var len=Animal.length
println("the size of the array is:"+len)
}
}
println("1.add, 2.remove, 3.display, 4.length, 5.exit");
println("enter the choice");
s=scala.io.StdIn.readInt();
}
```

```
}
}
```

Output:

```
1.add, 2.remove, 3.display, 4.length, 5.exit
enter the choice
1
At which position would you like to add the element
2
enter the element
Fox
1.add, 2.remove, 3.display, 4.length, 5.exit
enter the choice
2
enter the position of the element
4
1.add, 2.remove, 3.display, 4.length, 5.exit
enter the choice
3
The elements of the Array:
Lion
Tiger
Fox
Elephant
Leopard
1.add, 2.remove, 3.display, 4.length, 5.exit
enter the choice
4
the size of the array is:5
1.add, 2.remove, 3.display, 4.length, 5.exit
enter the choice
5
```

Classes and Objects :

1) Define a class CurrentAccount (accNo, name, balance, minBalance). Define appropriate constructors and operations withdraw(), deposit(), viewBalance(). Create an object and perform operations.

Ans:

```
import scala.io.StdIn._
import scala.util._
class CurrentAccount(var ano:Int,var nam:String,var minBal:Float)
{
var  accNo:Int=ano
var name:String=nam
var balance:Float=minBal
var minBalance:Float=minBal

def withdraw()
{
println("Enter the amount to be withdraw:")
var amt:Float=scala.io.StdIn.readFloat()
if((balance-amt)>=minBalance)
{
println("Balance withdraw successfully:")
balance=balance-amt
println("Remaining Balance="+balance)
}
else
{
println("you can only withdraw amount greater than minBalance
i.e."+minBalance)
}
}

def deposit()
{
println("Balance before deposit is="+balance)
println("Enter the amount to deposit:")
var amt=scala.io.StdIn.readFloat()
balance=balance+amt
println("Balance after deposit="+balance)
```

```
}

def viewBalance()
{
println("Account Balance="+balance)
}
}
object CurrentAccountDemo
{
def main(args:Array[String])
{
println("Create New Account For Customer:")
println("Enter the account number:")
var ano=scala.io.StdIn.readInt()
println("Enter the account holder name:")
var nam=scala.io.StdIn.readLine()
println("Enter the account minimum balance:")
var min=scala.io.StdIn.readFloat()

var obj=new CurrentAccount(ano,nam,min)
var op=4
do
{
println("1.withdraw")
println("2.deposit")
println("3.viewBalance")
println("4.exit")
println("Enter the operation you want to perform:")
op=scala.io.StdIn.readInt()
op match {
case 1 =>obj.withdraw()
case 2 =>obj.deposit()
case 3 =>obj.viewBalance()
case whoa =>println("Unexpected case:"+whoa.toString)
}
}while(op!=4);
}
}
```

Output:

Create New Account For Customer:
Enter the account number:
101
Enter the account holder name:
Akash
Enter the account minimum balance:
0
1.withdraw
2.deposit
3.viewBalance
4.exit
Enter the operation you want to perform:
2
Balance before deposit is=0.0
Enter the amount to deposit:
5000
Balance after deposit=5000.0
1.withdraw
2.deposit
3.viewBalance
4.exit
Enter the operation you want to perform:
1
Enter the amount to be withdraw:
200
Balance withdraw successfully:
Remaining Balance=4800.0
1.withdraw
2.deposit
3.viewBalance
4.exit
Enter the operation you want to perform:
3
Account Balance=4800.0
1.withdraw
2.deposit
3.viewBalance
4.exit
Enter the operation you want to perform:

2) Define a class Employee (id, name, salary). Define methods accept() and display(). Display details of employee having maximum salary.

Ans:

```
 class Employee {

var id: Int = 0
var name: String = _
var salary: Int = 0


def accept(): Unit = {
println("Enter employee's id")
id = scala.io.StdIn.readInt()
println("Enter employee's name")
name = scala.io.StdIn.readLine()
println("Enter employee's salary")
salary = scala.io.StdIn.readInt()
}

def display(): Unit = {
println("The employee id: " + id+ ", name: " +name + " and salary: " +salary )
}
}

object EmployeeObj {
def main(args: Array[String]): Unit = {
println("Enter the no of employees")
val emp_no = scala.io.StdIn.readInt()
val emp = new Array[Employee](emp_no)

for (i <- 0 until emp_no) {
  emp(i) = new Employee
  emp(i).accept()
  emp(i).display()
}

var max: Employee = emp(0)

for (j <- 0 until emp_no){
```

```
if( emp(j).salary > max.salary ){
max = emp(j)
}
}
println("The employee with maximum salary is: id: " + max.id+ ", name: "
+max.name + " and salary: " + max.salary )
}
}
```

Output:

```
Enter the no of employees
3
Enter employee's id
10
Enter employee's name
Akash
Enter employee's salary
100000
The employee id: 10, name: Akash and salary: 100000
Enter employee's id
11
Enter employee's name
Sohail
Enter employee's salary
50000
The employee id: 11, name: Sohail and salary: 50000
Enter employee's id
12
Enter employee's name
Sachin
Enter employee's salary
75000
The employee id: 12, name: Sachin and salary: 75000
The employee with maximum salary is: id: 10, name: Akash and salary: 100000
```

3) Create abstract class Order (id, description). Derive two classes PurchaseOrder& SalesOrder with members Vendor and Customer. Create object of each PurchaseOrder and SalesOrder. Display the details of each account.

Ans:

```
abstract class Order
{
var id:Int=101
var desc:String="sweets"
var price:Int=500
}
class PurchaseOrder extends Order
{
var sweet_name:String="jalebi"
def purchesinfo()
{
println("order id : " +id)
println("description : " +desc)
println("name : " +sweet_name)
}}
class SalesOrder extends Order
{
var customer:String="Rushi dixit"
var vendor:String="Aditya Mohol"
def salesinfo()
{
println("price:" +price)
println("customer_name : " +customer)
println("vendor_name : " + vendor)
}
}
object Demo3
{
def main(args:Array[String])
{
var obj=new PurchaseOrder()
var obj1=new SalesOrder()
obj.purchesinfo()
obj1.salesinfo()
```

```
}
}
```

Output:

```
order id : 101
description : sweets
name : jalebi
price : 500
customer_name : Rushi dixit
vendor_name : Aditya Mohol
```

4) Create abstract class Shape with abstract functions volume() and display(). Extend two classes Cube and Cylinder from it. Calculate volume of each and display it.

Ans:

```
abstract class shape
{
def volume
def display
}
class cube extends shape
{
var a: Int = 0
var v: Int = 0
def volume()
{
a = 4
v = a * a * a
}
def display()
{
println("Volume of cube is : " +v)
}
}
class cylinder extends shape
```

```
{
var a: Double = 0
var h: Double = 0
var v: Double = 0
def volume()
{
a = 4
h = 1
v = 3.14 * a * a * h
}
def display()
{
println("Volume of cylinder is : " +v)
}
}
object main
{
def main (args : Array[String]) =
{
var obj = new cube()
obj.volume()
obj.display()
var obj1 = new cylinder()
obj1.volume()
obj1.display()
}
}
```

Output:
Volume of cube is : 64
Volume of cylinder is : 50.24

5) Create class Project (id, name, location). Define parameterized constructor. Keep a count of each object created and display the details of each project.

Ans:

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

```
object SingletonObject{
var count:Int=0;
def countObject():Int={
count+=1
return count
}
}
class Project(var id1:Int,var nam:String,var loc:String)
{
var id:Int=id1
var name:String=nam
var location:String=loc
var cnt=SingletonObject.countObject()
println("Number of objects created="+cnt)
def display()
{
println(" "+id+" "+name+" "+location)
}
}
object ProjectDemo
{
def main(args:Array[String])
{
var ob1=new Project(1,"Gallery Management System" ,"Pune")
ob1.display()
var ob2=new Project(2,"Bus Management System","Mumbai")
ob2.display()
var ob3=new Project(3,"Hotel Management System","Delhi")
ob3.display()
}
}
```

Output:

```
Number of objects created=1
 1 Gallery Management System Pune
Number of objects created=2
 2 Bus Management System Mumbai
Number of objects created=3
 3 Hotel Management System Delhi
```

6) Define a class Sports (id, name, description, amount). Derive two classes Indoor and Outdoor. Define appropriate constructors and operations. Create an object and perform operations.

Ans:

```
class sports
{
var id:Int=0
var name:String="name"
var des:String="des"
var amt:Int=0
}
class indoors extends sports
{
def accept()
{
println("enter the indoor sports id:")
id=scala.io.StdIn.readInt()
println("enter the indoor sports name:")
name=scala.io.StdIn.readLine()
println("enter the indoor sports description:")
des=scala.io.StdIn.readLine()
println("enter the indoor sports amt:")
amt=scala.io.StdIn.readInt()
}
def display()
{
println("indoor sports id:"+id)
println("indoor sports name:"+name)
println("indoor sports description:"+des)
println("indoor sports amt:"+amt)
}
}
class outdoors extends sports
{
def accept()
{
println("enter the outdoor sports id:")
id=scala.io.StdIn.readInt()
println("enter the outdoor sports name:")
```

```
name=scala.io.StdIn.readLine()
println("enter the outdoor sports description:")
des=scala.io.StdIn.readLine()
println("enter the outdoor sports amt:")
amt=scala.io.StdIn.readInt()
}
def display()
{
println("outdoor sports id:"+id)
println("outdoor sports name:"+name)
println("outdoor sports description:"+des)
println("outdoor sports amt:"+amt)
}
}
object sport{
def main(args:Array[String])
{
var obj1=new indoors()
obj1.accept()
obj1.display()
var obj2=new outdoors()
obj2.accept()
obj2.display()
}
}
```

Output:

```
enter the indoor sports id:
101
enter the indoor sports name:
Chess
enter the indoor sports description:
Chess is one of the oldest and most popular board games.
enter the indoor sports amt:
100
indoor sports id:101
indoor sports name:Chess
indoor sports description: Chess is one of the oldest and most popular board games.
indoor sports amt:100
```

```
enter the outdoor sports id:
201
enter the outdoor sports name:
Cricket
enter the outdoor sports description:
Cricket is a bat-and-ball game.
enter the outdoor sports amt:
20000
outdoor sports id:201
outdoor sports name:Cricket
outdoor sports description: Cricket is a bat-and-ball game.
outdoor sports amt:20000
```

7) Design abstract class Employee with computeSal() as 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.

Ans:

```
object class7
{
abstract class employeee()
{
def Computesal(sal:Int)
}
class worker() extends employeee
{
var cVolume = 0
def Computesal(sal:Int)
{
println("Enter no of Hours worker work : ")
var h = scala.io.StdIn.readLine.toInt
cVolume = h * sal
println("Total Salary of Worker : "+cVolume)
}
}
class manager() extends employeee
```

```
{
var cVolume = 0
def Computesal(sal:Int)
{
println("Enter no of Month manager Work : ")
var h = scala.io.StdIn.readLine.toInt
cVolume = h*sal
println("Total salary of manager : "+cVolume)
}
}
def main(args: Array[String])
{
println("Enter 1 for worker : ")
println("Enter 2 for Manager : ")
//var n = readInt()
var n = scala.io.StdIn.readLine.toInt
if(n == 1)
{
println("-------Worker------ ")
var s = new worker()
println("Enter name of worker : ")
var name = scala.io.StdIn.readLine.toString
println("Enter Wage of worker as per Hour : ")
var a = scala.io.StdIn.readLine.toInt
println("Name of Worker : "+name)
s.Computesal(a)
}
else if(n == 2)
{
println("-------Manager------- ")
var s = new manager()
println("Enter Name of Manager : ")
var name = scala.io.StdIn.readLine.toString
println("Enter Wage of Manager per Month : ")
var a = scala.io.StdIn.readLine.toInt
s.Computesal(a)
println("Name of Manager : "+name)
}
}
}
```

Output:

```
Enter 1 for worker :
Enter 2 for Manager :
1
.....Worker.....
Enter name of worker :
Akash
Enter Wage of worker as per Hour :
500
Name of Worker : Akash
Enter no of Hours worker work :
10
Total Salary of Worker : 5000
```

List :

1) Create Lists using five different methods(Lisp style , Java style, fill, range and tabulate methods).

Ans:

```
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)
}
}
```

Output:

```
Scala List:
Lisp style:
List(101, 226, 30)
Java style:
List(1, 2, 3, 4, 5, 6, 7)
Range List:
List(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14)
```

List(0, 2, 4, 6, 8, 10, 12, 14, 16, 18)
Uniform List:
List(PPL, PPL, PPL)
Tabulated List:
List(0, 1, 8, 27, 64)

2) Create two Lists and Merge it and store the sorted in ascending order.

Ans:

```
import scala.collection.immutable._
object MainObject{
def main(args:Array[String]){
var list1 = List(1,8,5,6,9,58,23,15,4)
var list2 = List(88,100)
print("Elements in list 1: ")
list1.foreach((element:Int) => print(element+" "))
print("\nElements in list 2: ")
list2.foreach((element:Int) => print(element+" "))
var list3 = list1 ++ list2
print("\nElement after merging list1 and list2: ")
list3.foreach((element:Int)=>print(element+" "))
var list5 = list3.sorted
print("\nElements in ascending order of list: ")
list5.foreach((element:Int)=>print(element+" "))
}
}
```

Output:

```
Elements in list 1: 1 8 5 6 9 58 23 15 4
Elements in list 2: 88 100
Element after merging list1 and list2: 1 8 5 6 9 58 23 15 4 88 100
Elements in ascending order of list: 1 4 5 6 8 9 15 23 58 88 100
```

3) Create a list of integers divisible by 3 from List containing numbers from 1 to 50.

Ans:

```
object List_int{
def main(args:Array[String])
{
val x=List.range(1,50)
println("Number divided by 3")
for
{
i<-x
if i%3==0
}
println(i)
}
}
```

Output:

```
Number divided by 3
3
6
9
12
15
18
21
24
27
30
33
36
39
42
45
48
```

4) Create a list of even numbers up to 10 and calculate its product.

Ans:

```
object Scala_List
{
def main(args: Array[String]): Unit =
{
val nums = List(1, 2, 3, 4 , 5 , 6 , 7 , 8 , 9 , 10)
println("Original list:")
println(nums)
val even_nums = nums.filter(_ % 2 ==0)
println("Even number of the list:")
println(even_nums)
val result = even_nums.product
println("product: ")
println(result)
}
}
```

Output:

```
Original list:
List(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
Even number of the list:
List(2, 4, 6, 8, 10)
product:
3840
```

5) Write a program to create list with 10 members using function 3n2+4n+6.

Ans:

```
object List
{
def main(args: Array[String])
{
var l = List.tabulate(10)(n=>3*n*n+4*n+6)
println("using tabulate = " +l)
}
}
```

Output:

```
using tabulate = List(6, 13, 26, 45, 70, 101, 138, 181, 230, 285)
```

6) Write a program to create a list of 1 to 100 numbers. Create second list from first list selecting numbers multiple of 10.

Ans:

```
object List
{
def main(args: Array[String])
{
var l = List.range(1,101)
var k = l.filter(_%10==0)
println(l)
println("Multiple of 10")
println(k)
}
}
```

Output:

```
List(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,
24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44,
45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65,
```

66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100)
Multiple of 10
List(10, 20, 30, 40, 50, 60, 70, 80, 90, 100)

7) Create a list of 50 members using function 2n+3. Create second list excluding all elements multiple of 7.

Ans:

```
object list
{
def main(args:Array[String])
{
val list1 = List.tabulate(50)(n=>(2*n) + 3)
println("Answer : ")
list1.foreach((element:Int) => print(element+" "))

val list2 = List.tabulate(50)(n=>(2*n) + 3).filterNot(i=>i%7==0)
println("\nAfter excluding all elements multiple of 7:" )
list2.foreach((element:Int) => print(element+" "))
}
}
```

Output:

```
Answer :
3 5 7 9 53 55 57 59 61 63 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99 101
After excluding all elements multiple of 7:
3 5 9 11 13 15 17 19 23 25 27 29 31 33 37 39 41 43 45 47 51 53 55 57 59 61 65 67 69 71 73 75 79 81 83 85 87 89 93 95 97 99 101
```

Map :

1) Write a user defined functions to convert lowercase letter to uppercase and call the function using Map.

Ans:

```
object map1
{
def main(args:Array[String])
{
println("Enter the string :")
val lower = scala.io.StdIn.readLine()
val upper = lower.map(c=>c.toUpper)
println("The Upper case string is : "+upper)
}
}
```

Output:

```
Enter the string :
hello world
The Upper case string is : HELLO WORLD
```

2) Write a program to create map with Rollno and FirstName. Print all student information with same FirstName.

Ans:

```
object Slip1
{
def main(a:Array[String])
{
var map=Map(1->"mayuri",2->"pooja",3->"pooja",4->"mayuri",5->"puja");
for((k1,v1) <- map)
{
for((k2,v2)<-map)
{
if(v1==v2 && k1!=k2)
{
```

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

Output:

```
Roll No: 1 FirstName:mayuri
Roll No: 2 FirstName:pooja
Roll No: 3 FirstName:pooja
Roll No: 4 FirstName:mayuri
```

Set :

1) Write a program to create two sets and find common elements between them.

Ans:

```
import scala.io.StdIn.{readInt}
import scala.collection.mutable.Set
object Set_p2{
def main(args:Array[String]){
val set1=Set[Int]()
println("how many numbers enter in a set :")
var n=readInt()
println("Enter number of elements in set :")
for(i<-0 to n-1){
set1 += readInt()
}
println("Largest element of the set is: "+set1.max)
println("Smallest element of the set is: "+set1.min)
}
}
```

Output:

```
how many numbers enter in a set :
3
Enter number of elements in set :
2
9
5
Largest element of the set is: 9
Smallest element of the set is: 2
```

2) Write a program to display largest and smallest element of the Set.

Ans:

```
import scala.io.StdIn.{readInt}
import scala.collection.mutable.Set
object Set_p1{
```

```
def main(args:Array[String]){
val set1=Set[Int]()
val set2=Set[Int]()
println("how many numbers enter in a set 1 :")
var n=readInt()
println("Enter number of elements in set :")
for(i<-0 to n-1){
set1 += readInt()
}
println("how many numbers enter in a set 2 :")
n=readInt()
println("Enter number of elements in set :")
for(i<-0 to n-1){
set2 += readInt()
}
println("Common elements between set1 and set2 are: "+set1.&(set2))

println("Common elements between set1 and set2 are: "+set1.intersect(set2))

}
}
```

Ans:

```
how many numbers enter in a set 1 :
2
Enter number of elements in set :
4
6
how many numbers enter in a set 2 :
4
Enter number of elements in set :
7
4
6
3
Common elements between set1 and set2 are: HashSet(4, 6)
Common elements between set1 and set2 are: HashSet(4, 6)
```

3) Write a program to merge two sets and calculate product and average of all elements of the Set.

Ans:

```
object set {
def main(args: Array[String]): Unit = {
val IntSet1 = Set(1, 2, 3, 4)
println("Set 1 is = "+IntSet1)
val IntSet2 = Set(5, 6, 7)
println("Set 2 is = "+IntSet2)
val MergeSet = IntSet1 ++ IntSet2
println("Set after Merging Two Sets = "+MergeSet)
val ProductOfSet = MergeSet.product
println("Product of the Merge Set is = "+ProductOfSet)
val LengthOfSet = MergeSet.size
println("Length of Merge Set = "+LengthOfSet)
val SumOfSet = MergeSet.sum
println("Sum of the Merge Set = "+SumOfSet)
val AverageOfSet = SumOfSet/LengthOfSet
println("Average of the Merge Set is = "+AverageOfSet)
}
}
```

Output:

```
Set 1 is = Set(1, 2, 3, 4)
Set 2 is = Set(5, 6, 7)
Set after Merging Two Sets = HashSet(5, 1, 6, 2, 7, 3, 4)
Product of the Merge Set is = 5040
Length of Merge Set = 7
Sum of the Merge Set = 28
Average of the Merge Set is = 4
```