1. Program on R Data types

AIM:

Write a program to illustrate the datatypes in R programming

DESCRIPTION:

Basic data types in R can be divided into the following types:
numeric - (10.5, 55, 787)
integer - (1L, 55L, 100L, where the letter "L" declares this as an integer)
complex - (9 + 3i, where "i" is the imaginary part)
character (a.k.a. string) - ("k", "R is exciting", "FALSE", "11.5")
logical (a.k.a. boolean) - (TRUE or FALSE)
We can use the class() function to check the data type of a variable:

PROGRAM:

```
v <- TRUE
  print(class(v))
  v <- 23.5
  print(class(v))
    v <- 2L
    print(class(v))
    v <- 2+5i
    print(class(v))
    v <- "TRUE"
    print(class(v))
    v <- charToRaw("Hello")
    print(class(v))
```

Output:

```
[1] "logical"
[1] "numeric"
[1] "integer"
[1] "complex"
[1] "character"
[1] "raw"
```

RESULT:
The program is executed sucessfully without any errors.

2.Program on R Objects

AIM:
Write a program to demonstrate objects in R programming

DESCRIPTION:

There are 5 basic types of objects in the R language:
Vectors

Atomic vectors are one of the basic types of objects in R programming. Atomic vectors can store homogeneous data types such as character, doubles, integers, raw, logical, and complex. A single element variable is also said to be vector.

Lists

List is another type of object in R programming. List can contain heterogeneous data types such as vectors or another lists.

Matrices

To store values as 2-Dimensional array, matrices are used in R. Data, number of rows and columns are defined in the matrix() function.

Factors

Factor object encodes a vector of unique elements (levels) from the given data vector.

Arrays

array() function is used to create n-dimensional array. This function takes dim attribute as an argument and creates required length of each dimension as specified in the attribute.

PROGRAM:

```
# Create a vector.
apple <- c('red','green',"yellow")
print(apple)

# Get the class of the vector.
print(class(apple))

# Create a list.
list1 <- list(c(2,5,3),21.3,sin)

# Print the list.
print(list1)

# Create a matrix.
M = matrix( c('a','a','b','c','b','a'), nrow = 2, ncol = 3, byrow = TRUE)
print(M)

# Create an array.
a <- array(c('green','yellow'),dim = c(3,3,2))
print(a)
```

Output:

```
[1] "red"    "green"  "yellow"
[1] "character"
[[1]]
[1] 2 5 3

[[2]]
[1] 21.3

[[3]]
function (x)  .Primitive("sin")

[,1] [,2] [,3]
[1,] "a"  "a"  "b"
[2,] "c"  "b"  "a"
, , 1
```

```
[,1]     [,2]     [,3]
[1,] "green"  "yellow" "green"
[2,] "yellow" "green"  "yellow"
[3,] "green"  "yellow" "green"

, , 2

[,1]     [,2]     [,3]
[1,] "yellow" "green"  "yellow"
[2,] "green"  "yellow" "green"
[3,] "yellow" "green"  "yellow"
```

RESULT:

The program ios executed sucessfully without any error.

3. Program on Control Statements

AIM:
Write a program to illustrate the control statements in R programming

DESCRIPTION:

Control statements are expressions used to control the execution and flow of the program based on the conditions provided in the statements. These structures are used to make a decision after assessing the variable. In this article, we'll discuss all the control statements with the examples.
In R programming, there are 8 types of control statements as follows:
if condition
if-else condition
for loop
nested loops
while loop
repeat and break statement
return statement
next statement

PROGRAM:

a) Reverse of a given number using Repeat in R

```
n = as.integer(readline(prompt = "Enter a number :"))
s = 0

repeat {
r = n %% 10
s = s*10 + r
n = n %/% 10

if (n==0){
break}
}
```

```
print(paste("The reverse of a given number is :", s))
```

Output:

```
Enter a number :123
[1] "The reverse of a given number is : 321"
```

b) Sum of Digits of a given number using While loop

```
n = as.integer(readline(prompt = "Enter a number :"))
s = 0

while (n > 0) {
r = n %% 10
s = s + r
n = n %/% 10
}
print(paste("Sum of the digits is :", s))
```

Output:

```
Enter a number :213
[1] "Sum of the digits is : 6"
```

C) print first n odd numbers in descending order in R using For Loop

print first n odd numbers in descending order in R

```
num = as.integer(readline(prompt = "Enter a number :"))

num = num - 1

for (i in num:0) {
a = 1 + i * 2
print(paste(a))
}
```

Output:

```
Enter a number :12

[1] "23"
[1] "21"
[1] "19"
[1] "17"
[1] "15"
[1] "13"
[1] "11"
[1] "9"
[1] "7"
```

[1] "5"

[1] "3"

[1] "1"

RESULT:

The program is executed sucessfully without any error.

4. R Programming using Functions

AIM:
Write a program to demonstrate functions in R programming

DESCRIPTION:
A function is a set of statements organized together to perform a specific task. R has a large number of in-built functions and the user can create their own functions.
In R, a function is an object so the R interpreter is able to pass control to the function, along with arguments that may be necessary for the function to accomplish the actions.
The function in turn performs its task and returns control to the interpreter as well as any result which may be stored in other objects.

PROGRAM:

a) Built in Functions

```
# Create a sequence of numbers from 32 to 44.
print(seq(32,44))

# Find mean of numbers from 25 to 82.
print(mean(25:82))

# Find sum of numbers frm 41 to 68.
print(sum(41:68))
```

Output:

```
[1] 32 33 34 35 36 37 38 39 40 41 42 43 44
[1] 53.5
[1] 1526
```

b)User Defined Functions

```
# Create a function to print squares of numbers in sequence.
new.function <- function(a) {
for(i in 1:a) {
b <- i^2
print(b)
}
}

# Call the function new.function supplying 6 as an argument.
new.function(6)
```

Output:

[1] 1
[1] 4
[1] 9
[1] 16
[1] 25
[1] 36

RESULT:
The program is executed sucessfully without any error.

5.Reading Data and Writing Data

AIM:

Write a R program to demonstrate Reading Data and Writing Data

PROGRAM:

a) Reading the data from CSV

> getwd()

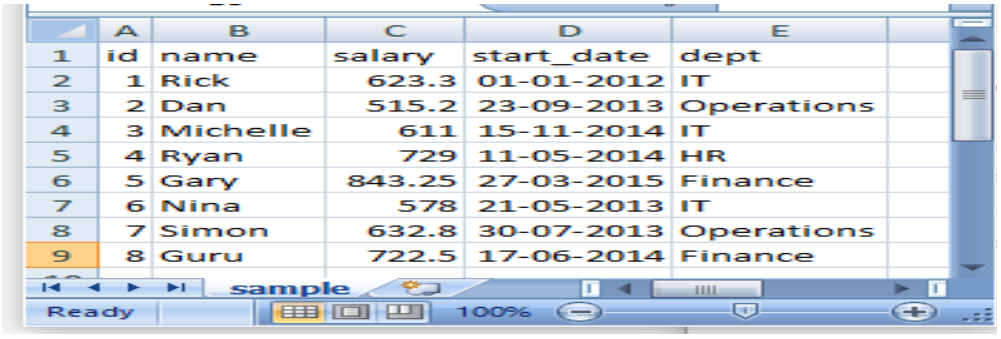
[1] "C:/Users/LENOVO/Documents"

> setwd("d:/")

> print(getwd())

[1] "d:/"

Sample.CSV



	A	B	C	D	E
1	id	name	salary	start_date	dept
2	1	Rick	623.3	01-01-2012	IT
3	2	Dan	515.2	23-09-2013	Operations
4	3	Michelle	611	15-11-2014	IT
5	4	Ryan	729	11-05-2014	HR
6	5	Gary	843.25	27-03-2015	Finance
7	6	Nina	578	21-05-2013	IT
8	7	Simon	632.8	30-07-2013	Operations
9	8	Guru	722.5	17-06-2014	Finance

sample
Ready
100%

> data <- read.csv("sample.csv")

> print(data)

id name salary start_date dept

1 1 Rick 623.30 01-01-2012 IT

2 2 Dan 515.20 23-09-2013 Operations

3 3 Michelle 611.00 15-11-2014 IT

4 4 Ryan 729.00 11-05-2014 HR

5 5 Gary 843.25 27-03-2015 Finance

6 6 Nina 578.00 21-05-2013 IT

7 7 Simon 632.80 30-07-2013 Operations

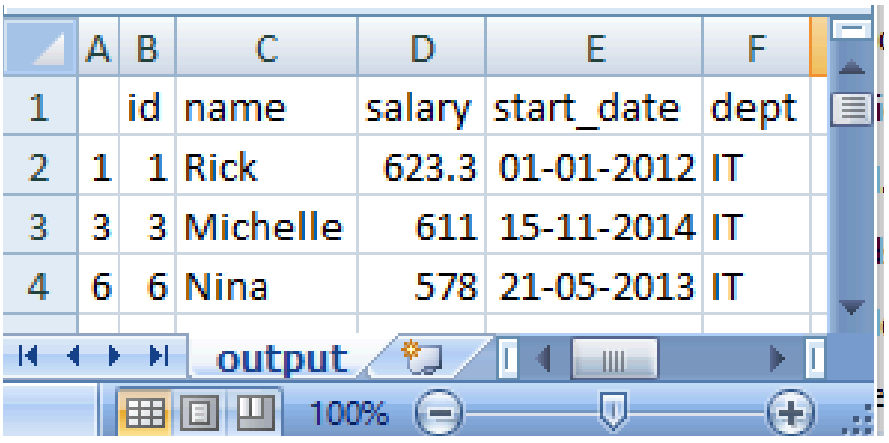
8 8 Guru 722.50 17-06-2014 Finance

b.) Writing the data using R Program

```
>data<- read.csv("sample.csv")

#Getting details of those peoples who joined in Department IT
> details <- subset(data,dept=="IT")
# Writing filtered data into a new file.
>write.csv(details,"output.csv")
>new_details<- read.csv("output.csv")
>print(new_details)
```

```
X id  name salary start_date dept
1 1  1   Rick  623.3 01-01-2012   IT
2 3  3 Michelle  611.0 15-11-2014   IT
3 6  6   Nina  578.0 21-05-2013   IT
```



OUTPUT:

The program is execute sucessfully without any error.

5. Loop Functions

AIM:

Write a program to demonstrate Loop functions in R programming

DESCRIPTION:

There are three types of loop in R programming:

For Loop

While Loop
Repeat Loop

For Loop in R

It is a type of control statement that enables one to easily construct a loop that has to run statements or a set of statements multiple times. For loop is commonly used to iterate over items of a sequence. It is an entry controlled loop, in this loop the test condition is tested first, then the body of the loop is executed, the loop body would not be executed if the test condition is false.

While Loop in R

It is a type of control statement which will run a statement or a set of statements repeatedly unless the given condition becomes false. It is also an entry controlled loop, in this loop the test condition is tested first, then the body of the loop is executed, the loop body would not be executed if the test condition is false.

Repeat Loop in R

It is a simple loop that will run the same statement or a group of statements repeatedly until the stop condition has been encountered. Repeat loop does not have any condition to terminate the loop, a programmer must specifically place a condition within the loop's body and use the declaration of a break statement to terminate this loop. If no condition is present in the body of the repeat loop then it will iterate infinitely.

PROGRAM:

```
# R program to demonstrate the use of for loop

# using for loop
for (val in 1: 5)
{
# statement
print(val)
}
```

Output:

```
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
```

```
# R program to demonstrate the use of while loop

val = 1

# using while loop
while (val <= 5)
{
# statements
print(val)
val = val + 1
}
```

Output:

```
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
```

```
# R program to demonstrate the use of repeat loop

val = 1

# using repeat loop
repeat
{
# statements
print(val)
val = val + 1

# checking stop condition
if(val > 5)
{
# using break statement
# to terminate the loop
break
}
}
```

Output:

```
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
```

RESULT:

The program is executed sucessfully without any error.

7.Data Visualization

AIM:

Write a R program to illustrate Data Visualization

DESCRIPTION:

Data Visualization in R Programming Language

The popular data visualization tools that are available are Tableau, Plotly, R, Google Charts, Infogram, and Kibana. The various data visualization platforms have different capabilities, functionality, and use cases. They also require a different skill set. This article discusses the use of R for data visualization.

R is a language that is designed for statistical computing, graphical data analysis, and scientific research. It is usually preferred for data visualization as it offers flexibility and minimum required coding through its packages.

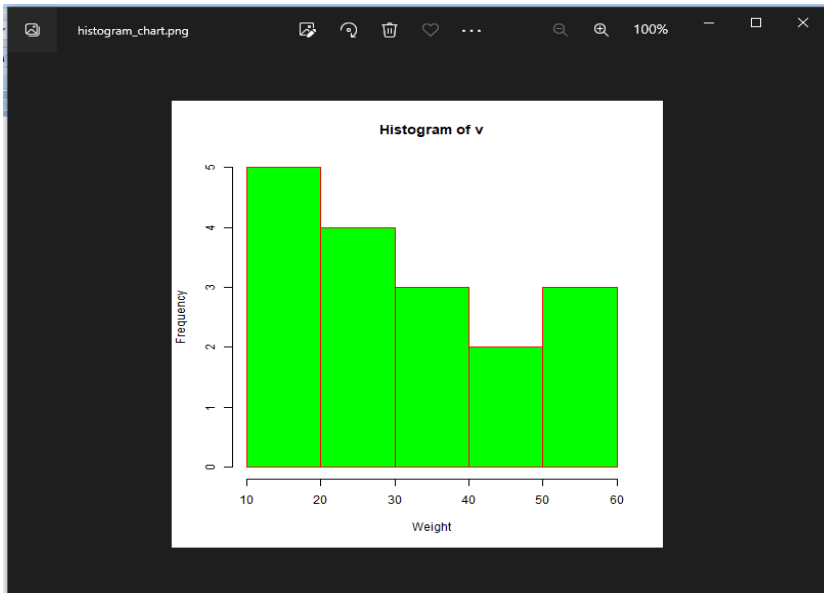
a)Bar chart

```
> # Creating the data for Bar chart
> H <- c(12,35,54,32,41)
> M<- c("Feb","Mar","Apr","May","Jun")
>
> # Giving the chart file a name
> png(file = "bar_properties.png")
>
```

```
> # Plotting the bar chart
> barplot(H,names.arg=M,xlab="Month",ylab="Revenue",col="Pink",
+         main="Revenue Bar chart",border="Yellow")
> # Saving the file
> dev.off()
```

b. Creating Histogram chart

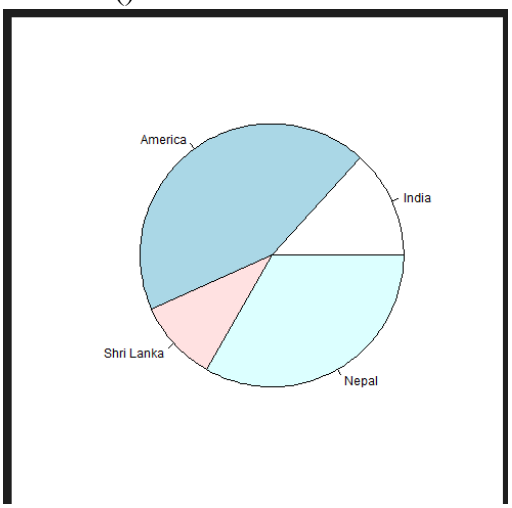
```
> # Creating data for the graph.
> v <-  c(12,24,25,26,16,38,36,21,13,55,17,39,42,43,10,58,60)
>
> # Giving a name to the chart file.
> png(file = "histogram_chart.png")
>
> # Creating the histogram.
> hist(v,xlab = "Weight",ylab="Frequency",col = "green",border = "red")
>
> # Saving the file.
> dev.off()
```



C. Creating Pie chart

```
# Creating data for the graph.
> x <- c(20, 65, 15, 50)
```

```
> labels <- c("India", "America", "Shri Lanka", "Nepal")
# Giving the chart file a name.
> png(file = "Country.jpg")
# Plotting the chart.
> pie(x,labels)
# Saving the file.
> dev.off()
```



RESULT:
The program is executed sucessfully without any error.

8. Correlation in R

AIM:
Write a program to demonstrate Correlation in R

PROGRAM:

```
> # R program to illustrate
> # pearson Correlation Testing
> # Using cor()
>
> # Taking two numeric
> # Vectors with same length
> x = c(1, 2, 3, 4, 5, 6, 7)
> y = c(1, 3, 6, 2, 7, 4, 5)
>
> # Calculating
> # Correlation coefficient
> # Using cor() method
> result = cor(x, y, method = "pearson")
>
> # Print the result
```

OUTPUT:

```
> cat("Pearson correlation coefficient is:", result)
Pearson correlation coefficient is: 0.5357143
```

RESULT:
The program is executed sucessfully without any error.

9. R Aggregate function

AIM:
Write a program to demonstrate Aggregate functions in R programming

PROGRAM:

```
> # create a dataframe with 4 columns
> data = data.frame(subjects=c("java", "python", "java", "java", "php", "php"),
                    id=c(1, 2, 3, 4, 5, 6),
                    names=c("manoj", "sai", "mounika", "durga", "deepika", "roshan"),
                    marks=c(89, 89, 76, 89, 90, 67))
>
> # display
> print(data)
subjects id   names marks
1     java  1   manoj    89
2   python  2     sai    89
3     java  3 mounika    76
4     java  4   durga    89
5      php  5 deepika    90
6      php  6  roshan    67
>
> # aggregate sum of marks with subjects
> print(aggregate(data$marks, list(data$subjects), FUN=sum))
Group.1   x
1    java 254
2     php 157
3  python  89
>
> # aggregate minimum of marks with subjects
> print(aggregate(data$marks, list(data$subjects), FUN=min))
Group.1  x
1    java 76
2     php 67
3  python 89
>
> # aggregate maximum of marks with subjects
> print(aggregate(data$marks, list(data$subjects), FUN=max))
Group.1  x
1    java 89
2     php 90
3  python 89
>
```

RESULT:
The program is executed sucessfully without any error.

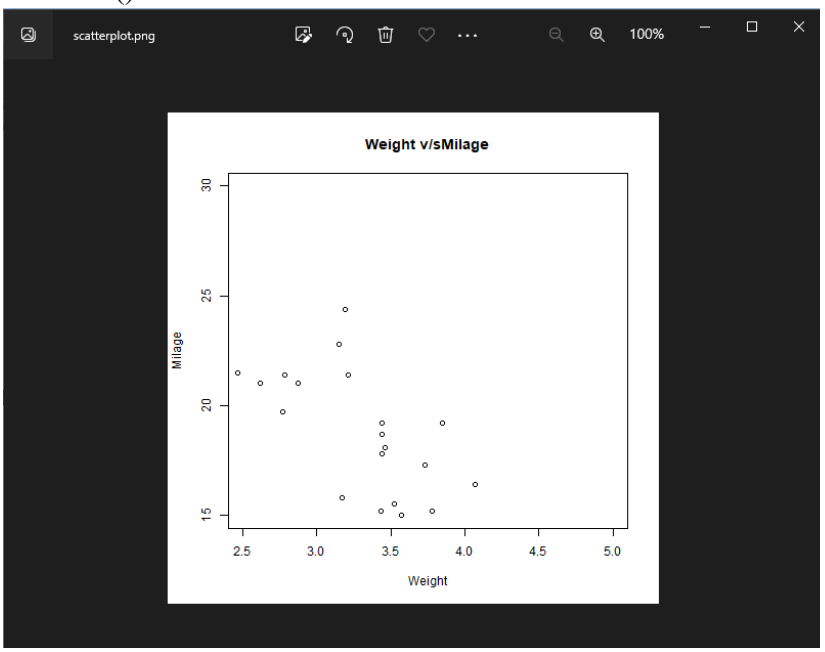
10. Scatter Plot using R

AIM:

Write a program to scatter a plot using R programming

PROGRAM:

```
> #Fetching two columns from mtcars
> data <-mtcars[,c('wt','mpg')]
> # Giving a name to the chart file.
> png(file = "scatterplot.png")
> # Plotting the chart for cars with weight between 2.5 to 5 and mileage between 15 and 30.
> plot(x = data$wt,y = data$mpg, xlab = "Weight", ylab = "Milage", xlim = c(2.5,5), ylim = c(15,30), main = "Weight v/sMilage")
> # Saving the file.
> dev.off()
```



RESULT:

The program is executed sucessfully without any error.