1.Write R program for different data structures in R.

Program:

```
# Numeric data type
numeric_var <- 42
cat("Numeric Variable:", numeric_var, "\n")

# Character data type
character_var <- "Hello, World!"
cat("Character Variable:", character_var, "\n")

# Logical data type
logical_var <- TRUE
cat("Logical Variable:", logical_var, "\n")

# Complex data type
complex_var <- 3 + 2i
cat("Complex Variable:", complex_var, "\n")

# Vector data type
numeric_vector <- c(1, 2, 3, 4, 5)
cat("Numeric Vector:", numeric_vector, "\n")
character_vector <- c("apple", "banana", "cherry")
cat("Character Vector:", character_vector, "\n")
logical_vector <- c(TRUE, FALSE, TRUE, FALSE)
cat("Logical Vector:", logical_vector, "\n")

# List data type
my_list <- list(name = "ashish", age = 30, hobbies = c("Reading", "Hiking"))
cat("List Variable:\n")
print(my_list)
```

```
# Data Frame data type
data_frame <- data.frame(
  Name = c("ashish", "raju", "bablu"),
  Age = c(25, 30, 35),
  Score = c(95, 88, 75)
)
cat("Data Frame Variable:\n")
print(data_frame)

# Matrix data type
my_matrix <- matrix(c(1,2,3,4,5,6), nrow = 2, ncol = 3)

cat("Matrix Variable:\n")
print(my_matrix)

# Factor data type
my_factor <- factor(c("Low", "Medium", "High", "Low"), levels = c("Low", "Medium",
"High"))
cat("Factor Variable:\n")
print(my_factor)
```

OUTPUT:

```
Numeric Variable: 42
Character Variable: Hello, World!
Logical Variable: TRUE
Complex Variable: 3+2i
Numeric Vector: 1 2 3 4 5
Character Vector: apple banana cherry
Logical Vector: TRUE FALSE TRUE FALSE
List Variable:
```

```
$name
[1] "ashish"
$age
[1] 30
$hobbies
[1] "Reading" "Hiking"
Data Frame Variable:
    Name Age Score
1 ashish  25    95
2   raju  30    88
3  bablu  35    75
Matrix Variable:
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
Factor Variable:
[1] Low    Medium High   Low
Levels: Low Medium High
```

2.Write R program that include variables,constants,datatypes.

Program:

```
logicvar = TRUE
cat(logicvar,"\n")
cat("the data type of variable logicvar is",class(logicvar), "\n\n")

 numvar = 111
cat(numvar,"\n")
cat("the data type of variable numvar is",class(numvar), "\n\n")

 intvar = 133L
cat(intvar,"\n")
cat("the data type of variable intvar is",class(intvar), "\n\n")

 complexvar= 2+3i
cat(complexvar,"\n")
cat("the data type of variable complexvar is",class(complexvar), "\n\n")

charvar = "R Programming"
 cat(charvar,"\n")
 cat("the data type of variable charvar is",class(charvar), "\n\n")

rawvar= charToRaw("R Programming")
cat(rawvar,"\n")
 cat("the data type of variable rawvar is",class(rawvar), "\n\n")

#built in Constants
cat(pi)
typeof(pi)
cat(letters)
cat(LETTERS)
cat(month.name)
```

cat(month.abb)

OUTPUT:

TRUE

the data type of variable logicvar is logical

111

the data type of variable numeric is numeric

133

the data type of variable intvar is integer

2+3i

the data type of variable complexvar is complex

R Programming

the data type of variable charvar is character

52 20 50 72 6f 67 72 61 6d 6d 69 6e 67

the data type of variable rawvar is raw

a b c d e f g h i j k l m n o p q r s t u v w x y z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

January February March April May June July August September October November December

Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec

3.Write R program that include different operators,control structures,default values for arguments,returning complex objects.

Program:

```
# Function to perform arithmetic operations
perform_operations = function(a, b = 5)
 {
    result = list(
                addition = a + b,
                subtraction = a - b,
                multiplication = a * b,
                division = a / b
              )
  return(result)
}

# Example usage of the function
values = perform_operations(10)  # Using default value for 'b'
print(values)

# Control structure example - if-else
if (values$addition > 10) {
  print("The addition result is greater than 10.")
} else {
  print("The addition result is less than or equal to 10.")
}
```

OUTPUT:

$addition

[1] 15

$subtraction

[1] 5$multiplication

[1] 50

$division

[1] 2

[1] "The addition result is greater than 10."

4.Write R program for quicksort implementation,binary search tree.

Program:

Quick Sort Program:

```
# Quicksort implementation in R
quicksort = function(arr) {
  if (length(arr) <= 1) {
    return(arr)
  }

  pivot = arr[1]
  less = arr[-1][arr[-1] <= pivot]
  greater = arr[-1][arr[-1] > pivot]

  return(c(quicksort(less), pivot, quicksort(greater)))
}

# Example usage
unsorted_array = c(3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5)
sorted_array = quicksort(unsorted_array)
cat("Sorted Array:", sorted_array, "\n")
```

OUTPUT:

Sorted Array: 1 1 2 33 4 55 5 69

Binary search tree program:

```
# Binary Search Tree implementation in R
# Define a node structure
createNode <- function(key) {
  return(list(
    key = key,
    left = NULL,
    right = NULL
  ))
}
# Insert a key into the BST
insert <- function(root, key) {
  if (is.null(root)) {
    return(createNode(key))
  }
  if (key < root$key) {
    root$left <- insert(root$left, key)
  } else if (key > root$key) {
    root$right <- insert(root$right, key)
  }
  return(root)
}
# In-order traversal of the BST
inorder <- function(root) {
  if (!is.null(root)) {
    inorder(root$left)
    cat(root$key, " ")
    inorder(root$right)
  }
}
```

```r
# Example usage
bst <- NULL
keys <- c(5, 3, 7, 2, 4, 6, 8)

# Insert keys into the BST
for (key in keys) {
  bst <- insert(bst, key)
}
# Print the in-order traversal of the BST
cat("In-order Traversal of BST:", "\n")
inorder(bst)
```

OUTPUT:

```
In-order Traversal of BST:
2  3  4  5  6  7  8
```

5.Write R program for calculating cumulative sums and products,minimum maximum and calcus.

Program:

```
my_vector <- c(2, 4, 6, 8, 10)
cumulative_sum <- cumsum(my_vector)
cat("Cumulative Sum:", cumulative_sum, "\n")

cumulative_product <- cumprod(my_vector)
cat("Cumulative Product:", cumulative_product, "\n")

min_value <- cummin(my_vector)
max_value <- cummax(my_vector)
cat("Cumulative Minimum Value:", min_value, "\n")
cat("Cumulative Maximum Value:", max_value, "\n")

derivative <- diff(my_vector)
cat("Derivative:", derivative, "\n")

integral <- cumsum(my_vector)
cat("Integral:", integral, "\n")
```

OUTPUT:

```
Cumulative Sum: 2 6 12 20 30
Cumulative Product: 2 8 48 384 3840
Minimum Value: 2
Maximum Value: 10
Cumulative Minimum Value: 2 2 2 2 2
Cumulative Maximum Value: 2 4 6 8 10
Derivative: 2 2 2 2
Integral: 2 6 12 20 30
```

6.Write a R program for finding stationarydistribution of markanovchains

```
# Install and load the markovchain package
install.packages("markovchain")
library(markovchain)

# Create a transition matrix for your Markov chain
# Example transition matrix for a simple 3-state Markov chain

transition_matrix <- matrix(c(0.7, 0.2, 0.1,
                              0.3, 0.6, 0.1,
                              0.2, 0.3, 0.5), nrow = 3, byrow = TRUE)
cat(transition_matrix)

# Create a markovchain object
markov_chain <- new("markovchain", states = c("State1", "State2", "State3"),
            transitionMatrix = transition_matrix, name = "Example Markov Chain")

# Compute the stationary distribution
stationary_distribution <- steadyStates(markov_chain)

# Print the stationary distribution
cat("Stationary Distribution:\n")
print(stationary_distribution)
```

OUTPUT:

```
0.7 0.3 0.2 0.2 0.6 0.3 0.1 0.1 0.5
        State1    State2    State3
[1,] 0.4722222 0.3611111 0.1666667
```

7.Write R program that include linear algebra operations on vectors and matrices.

```
# Create vectors
v1 <- c(1, 2, 3)
v2 <- c(4, 5, 6)
# Display vectors and matrices
cat("Vector 1:", v1, "\n")
cat("Vector 2:", v2, "\n")
# Addition of vectors
addition_result <- v1 + v2
cat("Vector Addition Result:", addition_result, "\n")

# Addition of vectors
subtraction_result <- v1 - v2
cat("Vector subtraction  Result:", subtraction_result, "\n")

# Scalar multiplication of a vector
scalar <- 2
scalar_multiplication_result <- scalar * v1
cat("Scalar Multiplication Result:", scalar_multiplication_result, "\n")

# Dot product of vectors
dot_product_result <- sum(v1 * v2)
cat("Dot Product Result:", dot_product_result, "\n")

# Create matrices
m1 <- matrix(c(1, 2, 3, 4, 5, 6), nrow = 2, byrow = TRUE)
m2 <- matrix(c(7, 8, 9, 10, 11, 12), nrow = 2, byrow = TRUE)

cat("Matrix 1:\n", m1, "\n")
cat("Matrix 2:\n", m2, "\n")
```

```
# Matrix addition
matrix_addition_result <- m1 + m2
cat("Matrix Addition Result:\n", matrix_addition_result, "\n")

# Matrix addition
matrix_subtaction_result <- m1 - m2
cat("Matrix subtraction Result:\n", matrix_subtaction_result, "\n")

# Matrix multiplication
matrix_multiplication_result <- m1 %*% t(m2)  # Transpose of m2 is used for multiplication
cat("Matrix Multiplication Result:\n", matrix_multiplication_result, "\n")
```

OUTPUT:

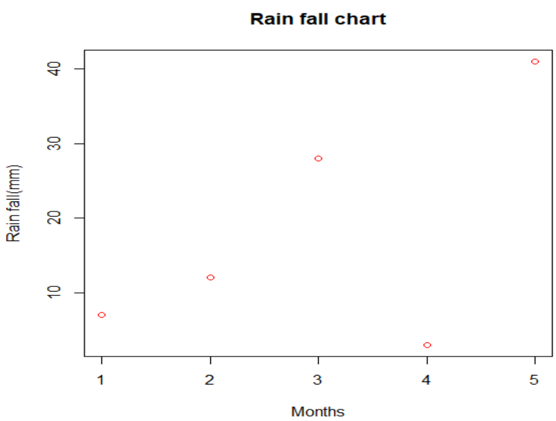
```
Vector 1: 1 2 3
Vector 2: 4 5 6
Vector Addition Result: 5 7 9
Vector subtraction  Result: -3 -3 -3
Scalar Multiplication Result: 2 4 6
Dot Product Result: 32
Matrix 1:
 1 4 2 5 3 6
Matrix 2:
 7 10 811 9 12
Matrix Addition Result:
 8 14 10 16 12 18
Matrix subtraction Result:
 -6 -6 -6 -6 -6 -6
Matrix Multiplication Result:
 50 122 68 167
```

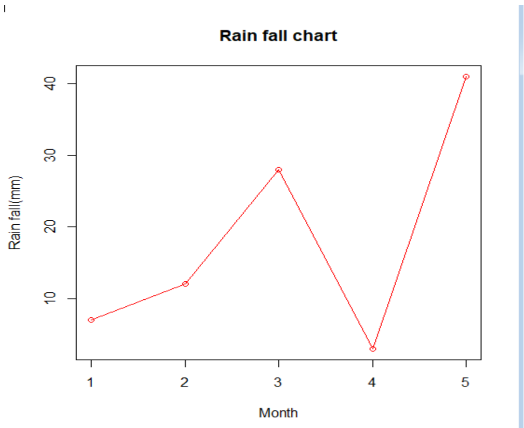
8.Write R program for any visual representation of an object with creating graphs using graphic fuctions: Plot() ,Hist() , Linearchart() , Pie() ,Boxplot(),Scatterplot().

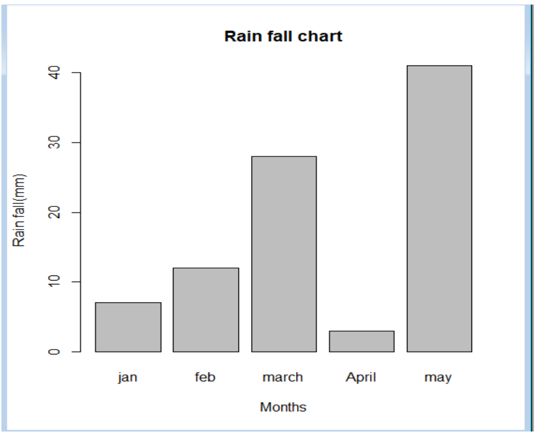
Program:

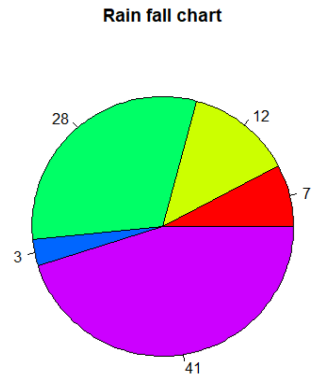
```
rainfall = c(7,12,28,3,41)

months = c("jan","feb","march","April","may")

plot(rainfall ,col = "red", xlab = "Months", ylab = "Rain fall(mm)", main = "Rain fall chart")


plot(rainfall , type = "o",col = "red", xlab = "Month", ylab = "Rain fall(mm)", main = "Rain fall chart")


barplot(rainfall , main = "Rain fall chart", xlab = "Months", ylab = "Rain fall(mm)",names.arg =months   )


pie(rainfall , labels = rainfall , main = "Rain fall chart",col = rainbow(length(rainfall)))

legend("topright",months , cex = 0.8, fill = rainbow(length(rainfall )))


rainfall_data = data.frame(Month = months , Rainfall = rainfall)

boxplot(Rainfall ~ Month, data = rainfall_data,main = "Rainfall Boxplot by Month",xlab = "Month", ylab = "Rainfall (mm)",col = "skyblue",border = "black")
```

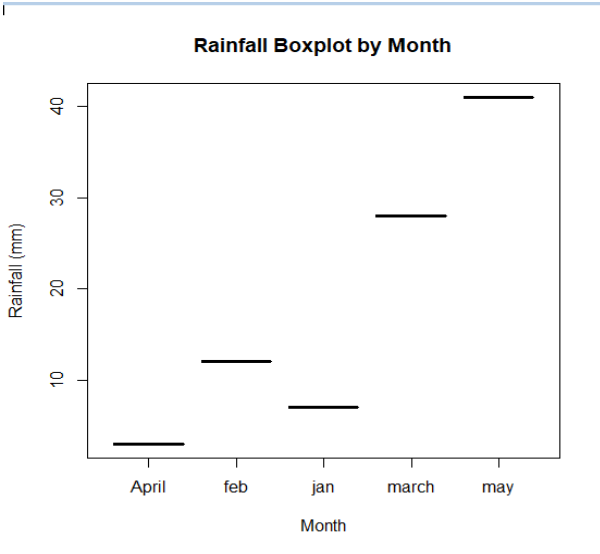
OUTPUT:






Rain fall chart
Rain fall(mm)
0
10
20
30
40
jan
feb
march
April
may
Months


Rain fall chart
28
12
7
3
41


Rainfall Boxplot by Month
Rainfall (mm)
10
20
30
40
April
feb
jan
march
may
Month

9.Write R program for with any data set containing data frame objects,indexing and subsetting data frames,and employ manipulating and analyzing data.

Program:

```
# Creating a sample dataframe
data = data.frame(
  Name = c("Ashish", "Divya", "Deeksha", "Deepak", "Rashmi"),
  Age = c(25, 30, 22, 35, 28),
  Gender = c("Male", "Female", "Female", "Male", "Female"),
  Score = c(78, 85, 62, 92, 80)
)

# Displaying the created dataframe
print("Original Data:")print(data)

# Indexing and subsetting the dataframe , Selecting rows where Age is greater than 25
subset_data <- data[data$Age > 25, ]

# Displaying the subset of data
print("\nSubset Data (Age > 25):")
print(subset_data)

# Manipulating data: Adding a new column
data$Grade <- ifelse(data$Score >= 80, "A", "B")

# Displaying the modified dataframe
print("\nData with Grade column added:")
print(data)

# Analyzing data: Summary statistics
summary(data)
```

OUTPUT:

```
[1] "Original Data:"
     Name Age Gender Score
1  Ashish  25   Male    78
2   Divya  30 Female    85
3 Deeksha  22 Female    62
4  Deepak  35   Male    92
5  Rashmi  28 Female    80

[1] "\nSubset Data (Age > 25):"
    Name Age Gender Score
2  Divya  30 Female    85
4 Deepak  35   Male    92
5 Rashmi  28 Female    80

[1] "\nData with Grade column added:"
     Name Age Gender Score Grade
1  Ashish  25   Male    78     B
2   Divya  30 Female    85     A
3 Deeksha  22 Female    62     B
4  Deepak  35   Male    92     A
5  Rashmi  28 Female    80     A

     Name                Age        Gender              Score
 Length:5           Min.   :22   Length:5           Min.   :62.0
 Class :character   1st Qu.:25   Class :character   1st Qu.:78.0
 Mode  :character   Median :28   Mode  :character   Median :80.0
                    Mean   :28                      Mean   :79.4
                    3rd Qu.:30                      3rd Qu.:85.0
                    Max.   :35                      Max.   :92.0
```

Grade

Length:5

Class :character

Mode :character