Practical no 1

a. Write an R program to implement expressions, assignment and decision making?

Codes:-

```
 x<-15
if(x>10)
  if(x<=20)
  {
    print("x is in between 10 and 20")
  }else
  {
 print("x is greater than 20")
  }else
  ( print("x is less than 10") )
```

b. Write am R Program to design and implement loops?

Codes:-

```
          a<-c(1:10)
for (y in a)
  {
  print(y*5)
  }
```

C. Write a R program to demonstrate the use of essential data structures in R?

Codes:-

```
result<-c("Hello World")
i<-1
repeat
{
 print(result)
      i=i+1
       if(i>6)
         break
}
```

Practical no:- 2

a. Write an R program to manage data and exhibit Operations on it using List data structures?

Codes:-

```
empId <-c (1,2,3,4)
empName <- c ("Debi","Sandeep","Shubham","Shiba")
numberofEmp <- 4

empList = list (“ID"=empId, "Names"=empName, "Total Staff"=numberofEmp)
print(empList)

cat ("Accessing name components using $ command\n")
print(empList$Names)
```

b. Write an R program to manage data and exhibit operations on it using data frames?

Codes:-

```
# R program to illustrate dataframe

# A vector which is a character vector
Name = c("Amiya", "Raj", "Asish")

# A vector which is a character vector
Language = c("R", "Python", "Java")

# A vector which is a numeric vector
Age = c(22, 25, 45)

# To create dataframe use data.frame command and
# then pass each of the vectors
# we have created as arguments
# to the function data.frame()
df = data.frame(Name, Language, Age)

print(df)
```

c. Write an R program to demonstrate the use of:

Codes:-

1. User define function;

Codes:-

```
add_num <- function(a,b)
{
  sum_result <- a+b
  return(sum_result)
}

sum = add_num(32,34)

print(sum)

#example 2
```

```
Square = function(side=2){
  area = side + side
  return(area)
}
print (Square (6))
```

2. Built in Function-

Codes:-

```
# Create a vector of numbers
numbers <- c(10, 20, 30, 40, 50)

# Calculate the mean using the built-in mean() function
average <- mean(numbers)

# Print the result
print(average)
```

Practical no:-3

a. Write an R program to store and access string in R objects (vectors, matrix, arrays, data frames, and lists)

Codes:-

1. **Vector**

```
# A vector is the simplest container. It's just a single row of items.
    cat("--- 1. Strings in a Vector ---\n\n")

    # Create a vector of strings. We use c() which means "combine".
    string_vector <- c("apple", "banana", "cherry")

    # Print the whole vector to see what's inside.
    print("The string vector:")
    print(string_vector)

    # To get just one item, we use its position (index) in square brackets [].
    # R starts counting from 1.
cat("\nGetting the second item from the vector:", string_vector[2], "\n\n\n")
```

2. ***Matrix***

```
# A matrix is a grid of items, with rows and columns.
# All items in a matrix must be of the same type (e.g., all strings).
cat("--- 2. Strings in a Matrix ---\n\n")

# Create a 2x2 matrix of strings.
string_matrix <- matrix(
  c("R", "Python", "Java", "SQL"),
  nrow = 2, # We want 2 rows
  ncol = 2  # We want 2 columns
```

```
)

# Print the whole matrix.
print("The string matrix:")
print(string_matrix)

# To get an item, we need its row and column number: [row, column].
cat("\nGetting the item in row 1, column 2:", string_matrix[1, 2], "\n\n\n")
```

3. Data Frame

```
# A data frame is like a spreadsheet (e.g., Excel). It has columns with names.
# Each column can have a different type of data (e.g., one numbers, one strings).
cat("--- 3. Strings in a Data Frame ---\n\n")

# Create a data frame.
employee_data <- data.frame(
  name = c("Alice", "Bob"),
  job = c("HR", "Engineering")
)

# Print the whole data frame.
print("The data frame:")
print(employee_data)
```

4. List

```
# A list is the most flexible container. It can hold anything,
# including other vectors, matrices, or even other lists.
cat("--- 4. Strings in a List ---\n\n")

# Create a list.
project_info <- list(
  projectName = "Website Redesign",
  teamMembers = c("Eve", "Frank", "Grace")
)

# Print the whole list.
print("The list:")
print(project_info)

# Like data frames, we can get items from a list using the '$' sign.
cat("\nGetting the project name from the list:", project_info$projectName, "\n")
cat("Getting the team members from the list:\n")
print(project_info$teamMembers)
```

b. Write an R program to demonstrate use of various string manipulation functions.

Codes :-

```
#First, you need to install and load the stringr package.
```

```
#You only need to install it once.
install.packages("stringr")

# Load the package into your R session.
library(stringr)# Let's define some strings to work with.

string1 <- "hello world"
string2 <- "   R programming is fun!   "
sentence <- "The quick brown fox jumps over the lazy dog."

# --- 1. Concatenating (joining) strings with str_c() ---
cat("--- 1. Joining Strings ---\n")
first_name <- "John"
last_name <- "Doe"

# str_c() joins strings together.
full_name <- str_c(first_name, " ", last_name)
cat("Joined name:", full_name, "\n\n")




# --- 2. Finding the length of a string with str_length() ---

cat("--- 2. String Length ---\n")
# str_length() counts the number of characters.
len <- str_length(string1)
cat("The length of '", string1, "' is:", len, "\n\n", sep="")




# --- 3. Changing case with str_to_upper() and str_to_lower() ---

cat("--- 3. Changing Case ---\n")

# str_to_upper() makes everything uppercase.

upper_case <- str_to_upper(string1)

cat("Uppercase:", upper_case, "\n")

# str_to_lower() makes everything lowercase.

lower_case <- str_to_lower("HELLO")

cat("Lowercase:", lower_case, "\n\n")
```

Code:-# --- 4. Trimming whitespace with str_trim() ---

```
cat("--- 4. Trimming Whitespace ---\n")
# str_trim() removes whitespace from the beginning and end.
cat("Original string: '", string2, "'\n", sep="")
trimmed_string <- str_trim(string2)
cat("Trimmed string:  '", trimmed_string, "'\n\n", sep="")
# --- 5. Detecting if a pattern exists with str_detect() ---
cat("--- 5. Detecting Patterns ---\n")
# str_detect() checks if a string contains a certain pattern and returns TRUE or FALSE.
has_fox <- str_detect(sentence, "fox")
has_cat <- str_detect(sentence, "cat")
cat("Does the sentence contain 'fox'?", has_fox, "\n")
cat("Does the sentence contain 'cat'?", has_cat, "\n\n")
```

Code:-# --- 6. Extracting parts of a string with str_sub() ---

```
cat("--- 6. Extracting Substrings ---\n")
# str_sub() lets you pull out a piece of a string.
# Get characters from position 1 to 5.
substring <- str_sub(sentence, 1, 5)
cat("The first 5 characters are: '", substring, "'\n\n", sep="")
# --- 7. Replacing parts of a string with str_replace() ---
cat("--- 7. Replacing Patterns ---\n")
# str_replace() finds a pattern and replaces it with something new.
new_sentence <- str_replace(sentence, "fox", "cat")
cat("Original sentence:", sentence, "\n")
cat("New sentence:     ", new_sentence, "\n\n")
```

```
# --- 8. Splitting a string with str_split() ---

cat("--- 8. Splitting a String ---\n")

# str_split() breaks a string into a list of smaller strings based on a separator.

words <- str_split(string1, " ")

cat("Splitting '", string1, "' by space gives us:\n", sep="")

# The result is a list, so we access the first element with [[1]]



tolower(Str_1)
toupper(Str_1)
paste("Good", "morning", sep = " ")
strsplit(Str_1, "to")
```

```
strsplit(Str_1, " ")
length(Str_1)
nchar(Str_1)
substr(Str_1, 7, 13)
grep("hello", Str_1, ignore.case = TRUE)
```

Practical no:4

a. Write an R program to apply built-in statistical functions?

Codes:-

```
data <- c(12, 15, 14, 10, 18, 22, 17, 14, 19, 20)

data_mean <- mean(data)
cat("Mean:", data_mean, "\n")

data_median <- median(data)
cat("Median:", data_median, "\n")

data_sd <- sd(data)
cat("Standard Deviation:", data_sd, "\n")

data_var <- var(data)
cat("Variance:", data_var, "\n")

data_summary <- summary(data)
cat("Summary:\n")
print(data_summary)

data_quantiles <- quantile(data)
cat("Quantiles:\n")
print(data_quantiles)
```

b. Write an R program to demonstrate Linear and Multiple Regression analysis?

Codes:-

```
x <- c(2, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29)

y <- c(4, 5, 7, 10, 13, 15, 17, 18, 20, 22, 24, 26, 28, 30, 32)

linear_model <- lm(y ~ x)

cat("Linear Regression Model Summary:\n")

print(summary(linear_model))

x1 <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15)

x2 <- c(5, 7, 8, 6, 9, 11, 13, 12, 14, 16, 15, 17, 18, 20, 22)
```

```
y_multi <- c(10, 12, 14, 13, 16, 19, 21, 20, 23, 25, 27, 28, 30, 33, 35)

multiple_model <- lm(y_multi ~ x1 + x2)

cat("\nMultiple Regression Model Summary:\n")

print(summary(multiple_model))

cat("\nCoefficients for Linear Regression:\n")

print(coef(linear_model))

cat("\nCoefficients for Multiple Regression:\n")

print(coef(multiple_model))

cat("\nPredicted values from Linear Regression:\n")

print(predict(linear_model))

cat("\nPredicted values from Multiple Regression:\n")

print(predict(multiple_model))
```

Practical no:-5

A. Normal Distribution. [Hint: dnorm(), pnorm(), qnorm(), rnorm()].

Codes:-

```
#The Normal Distribution
cat("------ Normal Distribution Demonstration ------\n\n")
mean_val <- 70  # Example: Average test score
std_dev_val <- 10  # Example: Standard deviation of test scores

cat("Parameters: Mean = ", mean_val, ", Standard Deviation = ", std_dev_val, "\n\n")

# --- i. dnorm(): Probability Density Function ---
x_point <- 75
density <- dnorm(x = x_point, mean = mean_val, sd = std_dev_val)
cat("1. dnorm(): The density (height of the curve) at x = 75 is:", round(density, 4), "\n")

# --- ii. pnorm(): Cumulative Distribution Function ---
prob_less_than_85 <- pnorm(q = 85, mean = mean_val, sd = std_dev_val)
cat("2. pnorm(): The probability of a value being <= 85 is:", round(prob_less_than_85, 4),
"\n")

prob_greater_than_85 <- pnorm(q = 85, mean = mean_val, sd = std_dev_val, lower.tail =
FALSE)
cat("   (The probability of a value being > 85 is:", round(prob_greater_than_85, 4), ")\n")
```

```
# --- iii. qnorm(): Quantile Function ---
percentile_90 <- qnorm(p = 0.90, mean = mean_val, sd = std_dev_val)
cat("3. qnorm(): The value for the 90th percentile is:", round(percentile_90, 2), "\n")

# --- iv. rnorm(): Random Number Generation ---
num_samples <- 10
random_scores <- rnorm(n = num_samples, mean = mean_val, sd = std_dev_val)
cat("4. rnorm(): Ten random numbers from this distribution are:\n")
print(round(random_scores, 2))
```

B. Binomial Distribution: [Hint: dbinom(), pbinom(),qbinom(),rbinom()].

Codes:-

```
# --- Binomial Distribution ---
# Parameters for the binomial distribution
num_trials <- 10  # n
prob_success <- 0.5  # p (e.g., probability of heads in a coin toss)

# i. dbinom(): Probability mass function
prob_exactly_5_successes <- dbinom(5, size = num_trials, prob = prob_success)
print(paste("Binomial Distribution - Probability of exactly 5 successes:",
round(prob_exactly_5_successes, 4)))

# ii. pbinom(): Cumulative distribution function
prob_up_to_5_successes <- pbinom(5, size = num_trials, prob = prob_success)
print(paste("Binomial Distribution - Probability of up to 5 successes:",
round(prob_up_to_5_successes, 4)))

# iii. qbinom(): Quantile function
quantile_75th_binom <- qbinom(0.75, size = num_trials, prob = prob_success)
print(paste("Binomial Distribution - 75th percentile (number of successes):",
quantile_75th_binom))

# iv. rbinom(): Random number generation
random_binomial_numbers <- rbinom(3, size = num_trials, prob = prob_success)
print("Binomial Distribution - 3 random numbers of successes:")
print(random_binomial_numbers)
```

Practical no:-6

A. Write an R program to demonstrate various ways of performing Graphical analysis.[Hint: Plots, Special Plots, Storing Graphics].

Codes:-

```
# --- Section 1: Create Basic Plots ---

# 1. Scatter Plot

plot(x = iris$Sepal.Length[1:10],

    y = iris$Sepal.Width[1:10],
```

```
    main = "Scatter Plot of Iris Sepal Dimensions",
    xlab = "Sepal Length",
    ylab = "Sepal Width")


# --- Section 2: Storing a Graphic ---
# Step 1: Tell R you want to start creating a PNG file.
png("My_Iris_Boxplot.png")
# Step 2: Create the plot you want to save.
boxplot(Petal.Width ~ Species, data = iris,
        main = "Box Plot of Petal Width by Species",
        ylab = "Petal Width",
        col = c("red", "green", "blue"))  # Added some colors for each species
# Step 3: Tell R you are finished. This command closes and saves the file.
dev.off()


library(ggplot2)
# --- Section 1: Create Basic Plots with ggplot2 ---
# 1. Scatter Plot
ggplot(data = iris, aes(x = Sepal.Length, y = Sepal.Width)) +
  geom_point() +  # Adds the points to create the scatter plot
  labs(title = "Scatter Plot of Iris Sepal Dimensions",
       x = "Sepal Length",
       y = "Sepal Width")
```

```
# 2. Histogram

ggplot(data = iris, aes(x = Petal.Length)) +

  geom_histogram(binwidth = 0.25, fill = "lightblue", color = "black") +  # 'fill' is bar color, 'color' is border

  labs(title = "Histogram of Iris Petal Length",

       x = "Petal Length")




# 3. Box Plot

ggplot(data = iris, aes(x = Species, y = Petal.Width)) +

  geom_boxplot() +  # Adds the box plots

  labs(title = "Box Plot of Petal Width by Species",

       x = "Species",

       y = "Petal Width")


# --- Section 2: Storing a ggplot Graphic ---

# Step 1: Create the plot and assign it to a variable.

my_iris_plot <- ggplot(data = iris, aes(x = Species, y = Petal.Width, fill = Species)) +

  geom_boxplot() +

  labs(title = "Box Plot of Petal Width by Species",

       x = "Species",

       y = "Petal width") +

  theme(legend.position = "none")  # Hide the legend, as the x-axis labels are enough


# Step 2: Use ggsave() to save the plot stored in the variable.

ggsave("My_Iris_Boxplot_ggplot.png", plot = my_iris_plot, width = 6, height = 4)
```

Practical no:- 7

A. Write an R program to demonstrate OOP concepts, the construction and use of S3 class.

Codes:-

```
employee1<-list(name="peter", age=21, role="developer")

class(employee1)<-"Employee_info"

print.Employee_info<-function(obj){

  cat(obj$name,"\n")

  cat(obj$age,"years old\n")

  cat("role:",obj$role,"\n")

}

print(employee1)
```

Practical no:- 8

A. Write an R program to demonstrate data interface with CSV files [Hint:- Creating data for CSV, analyzing, writing CSV files].

Codes:-

```
# R Program to Demonstrate Interfacing with CSV Files

# --- Part 1: Creating Data for a CSV File ---

employee_data <- data.frame(

  EmployeeID = c(101, 102, 103, 104, 105),

  Name = c("John Smith", "Jane Doe", "Peter Jones", "Mary Williams", "David Brown"),

  Department = c("Sales", "Marketing", "HR", "Sales", "IT"),

  Salary = c(55000, 62000, 58000, 56000, 75000)

)

print("-- Original Data Frame in R ---")

print(employee_data)
```

```
# --- Part 2: Writing the Data Frame to a CSV File ---

write.csv(employee_data, "employees.csv", row.names = FALSE)

print("\n--- Data successfully written to employees.csv ---")


# --- Part 3: Reading and Analyzing the CSV File ---

read_employee_data <- read.csv("employees.csv")

print("\n--- Data Read from employees.csv ---")

print(read_employee_data)

print("\n--- Analysis of the CSV Data (Summary) ---")

summary(read_employee_data)

print("\n--- Structure of the Loaded Data ---")

str(read_employee_data)
```

B. Write an R program to work with spreadsheet (Excel) programs. [Hint:installing, loading, verifying, creating data for xlsx file].

Codes:-

```
# R Program to Demonstrate Working with Excel Files (.xlsx)

# --- Part 1: Installing and Loading the Required Package ---

library(openxlsx)

print("--- 'openxlsx' package loaded successfully ---")


# --- Part 2: Creating Data for the Excel File ---

product_inventory <- data.frame(

  ProductID = c("A-001", "A-002", "B-001", "B-002", "C-001"),

  ProductName = c("Laptop", "Mouse", "Keyboard", "Monitor", "Webcam"),
```

```
  StockQuantity = c(50, 250, 150, 75, 200),
  Price = c(1200.00, 25.50, 75.00, 300.00, 45.75)
)
print("\n--- Original Product Inventory Data Frame ---")
print(product_inventory)

# --- Part 3: Writing the Data Frame to an Excel File ---
write.xlsx(product_inventory, "inventory.xlsx")
print("\n--- Data successfully written to inventory.xlsx ---")

# --- Part 4: Reading and Verifying the Excel File ---
read_inventory_data <- read.xlsx("inventory.xlsx")
print("\n--- Data Read from inventory.xlsx ---")
print(read_inventory_data)
```

Practical no:- 9

A. write a R program to manage data using dplyr Package[Hint: group by(), codes:-

```
library(dplyr)
data <- data.frame(
  name = c("Alice", "Bob", "Charlie", "David", "Emily"),
  age = c(25, 30, 22, 28, 27),
  city = c("New York", "Los Angeles", "Chicago", "London", "Paris")
)
filtered_data <- filter(data, age > 25)
selected_data <- select(data, name, age)
```

```
arranged_data <- arrange(data, desc(age))

mutated_data <- mutate(data, is_adult = age >= 18)

summary_data <- summarize(data, mean_age = mean(age))

grouped_data <- group_by(data, city)

summarized_grouped_data <- summarize(grouped_data, mean_age = mean(age))

result <- data %>%

  filter(age > 25 & city == "New York") %>%

  select(name)
```

```
library(dplyr)

# 1. Select specific columns

mtcars %>%

  select(mpg, cyl, disp)

# 2. Filter rows based on conditions

mtcars %>%

  filter(cyl == 4)

# 3. Arrange rows by a specific column

mtcars %>%

  arrange(desc(mpg))

# 4. Create a new column

mtcars %>%

  mutate(mpg_per_cyl = mpg / cyl)

# 5. Group data and summarize

mtcars %>%
```

```
  group_by(cyl) %>%
  summarize(mean_mpg = mean(mpg))
# 6. Combine multiple operations using the pipe
mtcars %>%
  filter(cyl == 4) %>%
  select(mpg, hp) %>%
  arrange(desc(hp))



library(dplyr)
# Example using mtcars dataset
mtcars %>%
  group_by(cyl) %>%
  mutate(avg_mpg_per_cyl = mean(mpg)) %>%
  rename(num_cylinders = cyl) %>%
  arrange(desc(avg_mpg_per_cyl)) %>%
  filter(avg_mpg_per_cyl > 20) %>%
  select(num_cylinders, avg_mpg_per_cyl)
```

Practical no:-10

A. Write an R program to demonstrate various error messages in R Programming.

Codes:-

```
# 1. Undefined variable

try(print(x))  # Error: object 'x' not found

# 2. Wrong data type (string + number)

try("text" + 5)  # Error: non-numeric argument to binary operator

# 3. Index out of bounds

v <- c(1, 2, 3)

print(v[10])  # [1] NA

# 4. Function misuse

try(mean(1, 2, 3))  # Error: unused arguments (2, 3)

# 5. File not found

try(read.csv("nofile.csv"))

# Error in file(file, "rt") : cannot open the connection

# In addition: Warning message:

# In file(file, "rt") : cannot open file 'nofile.csv': No such file or director
```

B. Write an R program to implement Error Handling in R [Hint: warning(),stop(),try(), tryCatch(), CallingHandlers()]

Codes:-

```
# 1. warning()

testWarn <- function(x) {

  if (x < 0) warning("Negative input!")

  return(abs(x))
```

```
}
testWarn(-5)
# 2. stop()
testStop <- function(x) {
  if (x < 0) stop("Stop! Negative not allowed.")
  return(x)
}
try(testStop(-3))
# 3. try()
res <- try(log("a"), silent = TRUE)
if (inherits(res, "try-error")) cat("Handled with try():", res, "\n")
# 4. tryCatch()
safeAdd <- function(a, b) {
  tryCatch({
    result <- a + b   # This will error if types are incompatible
    print(result)
  }, error = function(e) {
    cat("Error:", e$message, "\n")
  })
}
safeAdd(5, "a")
# 5. withCallingHandlers()
withCallingHandlers({
  warning("This is a warning.")
}, warning = function(w) {
```

```
  cat("Warning handled:", w$message, "\n")
  invokeRestart("muffleWarning")
})
```