1. Write an R program Illustrate with if-else statement and how does it operate on vectors of variable length.

```
# creating a vector of variable length
my_variable <- c(-2,0,5,-3,2,6)

# apply if-else condition
result <- ifelse(my_variable > 0, "Positive","Negative")
result
```

2. Write an R program Illustrate with for loop and stop on condition, to print the error message.

```
for (i in 1:10) {
 if(i!=5){
        print(paste("Finished loop iteration No.",i))
 }
 #stop at i==5
 if(i == 5){
        stop("i is equal to 5")
 }
}
```

3. Write an R Program To find Factorial of given number using recursion.

```
fact_rec <- function(n)
{
 if(n == 0 || n == 1)
 {
 return(1)
 }else
 {
 return(n*fact_rec(n-1))
 }
}
print(paste("the factorial of number 5 is", fact_rec(5)))
```

5. Write an R Program Compute mean values for vector aggregates defined by factors tapply and sapply

```
# Sample data
data <- c(5, 10, 15, 20, 25, 30)
factor <- c("A", "A", "B", "B", "C", "C")

# Compute mean values using tapply
mean_tapply <- tapply(data, factor, mean)
print(mean_tapply)

# Compute mean values using sapply
mean_sapply <- sapply(split(data, factor), mean)
print(mean_sapply)
```

7. Write an R Program for implementing Quick Sort for Binary Search.

```
 # Function to perform Quick Sort
quick_sort <- function(arr)
 {
  if (length(arr) <= 1)
  {
    return(arr)
  }
  pivot <- arr[1]
  lesser <- arr[arr < pivot]
  greater <- arr[arr > pivot]
  return(c(quickSort(lesser), pivot, quickSort(greater)))
 }

 binary_search <- function(arr, target) {
  sorted_array <- sort(arr)
  low <- 1
  high <- length(sorted_array)
  while (low <= high) {
    mid <- (low + high) %/% 2
    if (sorted_array[mid] == target) {
      return(paste("Found", target, "at index", mid))
    } else if (sorted_array[mid] < target) {
      low <- mid + 1
    } else {
      high <- mid - 1
    }
  }
  return(paste(target, "not found in the array"))
 }
```

```
# Example usage:
my_array <- c(5,4,2,1,27,8,9)
target_value <- 9
sorted_array <- quick_sort(my_array)
sorted_array
result <- binary_search(my_array, target_value)
print(result)
```

8. Write an R Program Illustrate Reading & Writing Files.

```
#creating a file
Employee_name<-c("Robert","Sachin","Priya","Arun")
Salary<-c(9000,12000,18000,20000)
Employee_details<-data.frame(Employee_name,Salary,stringsAsFactors = FALSE)
Employee_details

#Writing A file
write_file<-write.table(Employee_details,"EMP.txt")
cat("Data succesfully writen into file","EMP.txt","\n")

#Reading A file
Read_file<-read.table("EMP.txt",header = TRUE)
Read_file
```

9. Write a R program for any visual representation of an object with creating graphs using graphic functions: Plot(),Hist(),Linechart(),Pie(),Boxplot(),Scatterplots().

```
#Scatter Plot
x <- c(1, 2, 3, 4, 5)
y <- c(2, 4, 6, 8, 10)
plot(x, y, main = "Scatter Plot", xlab = "X", ylab = "Y")
```

```
#Line Graph
x <- c(1, 2, 3, 4, 5)
y <- c(2, 4, 6, 8, 10)
plot(x, y, type = "l", main = "Line Graph", xlab = "X", ylab = "Y")
```

```
#Bar Chart
x <- c("A", "B", "C", "D", "E")
y <- c(10, 20, 30, 40, 50)
barplot(y, main = "Bar Chart", xlab = "Category", ylab = "Value")
```

```
#Histogram
x = c(1,2,3,4,2,3,1,5,6,1,8,8,9,1,2,10,10,10)
hist(x, main = "Histogram", xlab = "Value", ylab = "Frequency")


#Box Plot
x = c(1,2,3,4,5)
boxplot(x, main = "Box Plot", ylab = "Value")

#Pie Chart
x <- c(20, 30, 50)
labels <- c("A", "B", "C")
pie(x, labels, main = "Pie Chart")
```

12. Write an R Program to Find Mean, Mode & Median

```
# Function to find mode
find_mode <- function(x) {
tbl <- table(x)
 mode_values <- as.numeric(names(tbl[tbl == max(tbl)]))
 return(mode_values)
}

# Sample numeric vector
numeric_vector <- c(5, 7, 8, 3, 5, 2, 8, 7, 5, 6, 8)

# Mean
mean_value <- mean(numeric_vector)
cat("Mean:", mean_value, "\n")

# Mode
mode_values <- find_mode(numeric_vector)
cat("Mode:", ifelse(length(mode_values) > 1, paste(mode_values, collapse = ", "),
 mode_values), "\n")

# Median
median_value <- median(numeric_vector)
cat("Median:", median_value, "\n")
```

6. Write a R program for finding stationary distribution of markanov chains.

```
 # Load the markovchain package
 library(markovchain)
 # Define the transition matrix of the Markov chain
 # Replace this with your own transition matrix
```

```
P <- matrix(c(0.7, 0.3, 0.2, 0.8), nrow = 2, byrow = TRUE)
# Create a markovchain object
mc <- new("markovchain", states = c("State1", "State2"), transitionMatrix = P)
# Find the stationary distribution
stationary_distribution <- steadyStates(mc)
# Print the stationary distribution
cat("Stationary Distribution:")
print(stationary_distribution)
```