Q1. Create 2 vector for numeric data and display add, sub, mul, div.

```
vector1 <- c(10, 5, 2)

vector2 <- c(3, 8, 1)

cat("Vector 1:", vector1, "\n")

cat("Vector 2:", vector2, "\n")

vector_add <- vector1 + vector2

cat("Addition:", vector_add, "\n")

vector_sub <- vector1 - vector2

cat("Subtraction:", vector_sub, "\n")
```

Q2. Input vector for 10 students , sort and display in order.

```
student_names <- character(0)

for(i in 1:10){

  name <- readline(paste("Enter name", i, ": "))

  student_names <- c(student_names, name)

}

sorted_names <- sort(student_names)

cat("Student Names (Sorted):\n")

for(name in sorted_names){

  cat(name, "\n")

}
```

Q3. Create a list in a DS that has components of mixed data types.

```
my_list <- list(

  name = "Alice",

  age = 25,

  courses = c("R", "Python", "Statistics"),

  enrolled = TRUE

)


cat("Name:", my_list$name, "\n")

cat("Age:", my_list$age, "\n")

cat("Courses:", my_list$courses, "\n")

cat("Enrolled:", my_list$enrolled, "\n")
```

Q4. Fibonacci series

```
number_of_terms <- as.integer(readline("Enter the number of terms for the Fibonacci series: "))
fibonacci_sequence <- c(0, 1)
for (i in 3:number_of_terms) {
  next_term <- fibonacci_sequence[i - 1] + fibonacci_sequence[i - 2]
  fibonacci_sequence <- c(fibonacci_sequence, next_term)
}
cat("Fibonacci Series:", fibonacci_sequence, sep = " ")
```

Q5.Implement decision tree on credit card issue dataset(import from kaggle).

```
library(party)
iris_1=iris[sample(150),]
View(iris_1)
train=iris_1[1:100,]
test=iris_1[101:150,]
tree=ctree(Species~Petal.Length+Petal.Width,data=train)
plot(tree)
p=predict(tree,test)          p
table(p,test$Species)
accuracy=(17+16+16)/50*100
accuracy
```

Q7. Naïve Bayes on iris dataset.

```
library(datasets)
data(iris)
install.packages("caret")
library(caret)
set.seed(42)
trainindex <- createDataPartition(iris$Species, p = 0.8, list = FALSE)
train <- iris[trainindex,]
test <- iris[-trainindex,]
library(e1071)
model <- naiveBayes(Species~ ., data = train)
predictions <- predict(model, newdata = test)
library(caret)
```

Q8. Input two matrix and display addition.

```
matrix1 <- matrix(1:9, nrow = 3, byrow = TRUE)

matrix2 <- matrix(9:1, nrow = 3, byrow = TRUE)

cat("Matrix 1:\n")

print(matrix1)

cat("\nMatrix 2:\n")

print(matrix2)

addition_matrix <- matrix1 + matrix2

cat("\nAddition Matrix:\n")

print(addition_matrix)
```

Section-B

Q1. RandomForest on iris dataset.

```
library(randomForest)

iris_1=iris[sample(150),]

View(iris_1)

train=iris_1[1:100,]

test=iris_1[101:150,]

model=randomForest(Species~.,data=train)

plot(model)

p=predict(model,test)         p

table(p,test$Species)

accuracy= (17+16+16)/50*100

accuracy
```

Q7. Import iris dataset and display first three columns.

```
# Load the iris dataset

data("iris")

# Display the first three columns of the iris dataset

head(iris[, 1:3])
```

Q8. Import iris dataset and display first, 3rd and last column.

```
data("iris")

# Display the specified columns

iris_subset <- iris[, c(1, 3, ncol(iris))]

# Selecting first, third, and last column

print(iris_subset)
```

Q3. Implement linear regression on iris dataset.

```
require("datasets")

data("iris")

str(iris)

head(iris)

Y<- iris[,"Sepal.Width"]

X<- iris[,"Sepal.Length"]

xycorr<- cor(Y,X, method="pearson")

xycorr

plot(Y~X, col=X)

model1<- lm(Y~X)

model1

plot(Y~X, col=X)

abline(model1, col="blue", lwd=3)
```

Q5. Implement logistic regression on the iris dataset.

```
data(iris)

logistic_model <- glm(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width, data = iris, family = binomial)

iris$predicted_prob <- predict(logistic_model, type = "response")

plot(iris$Sepal.Length, iris$predicted_prob, main = "Logistic Regression Curve for Setosa",

   xlab = "Sepal Length", ylab = "Predicted Probability", col = iris$Species, pch = 19)
```

Q6. Implement Apriori algorithm.

```
library(arules)

library(arulesViz)

library(RColorBrewer)

# import dataset

data("Groceries")

rules <- apriori(Groceries,

          parameter = list(supp = 0.01, conf = 0.2))

inspect(rules[1:10])

arules::itemFrequencyPlot(Groceries, topN = 20,

             col = brewer.pal(8, 'Pastel2'),

             main = 'Relative Item Frequency Plot',

                type = "relative",

             ylab = "Item Frequency (Relative)")
```