Name: Varun Sudhir

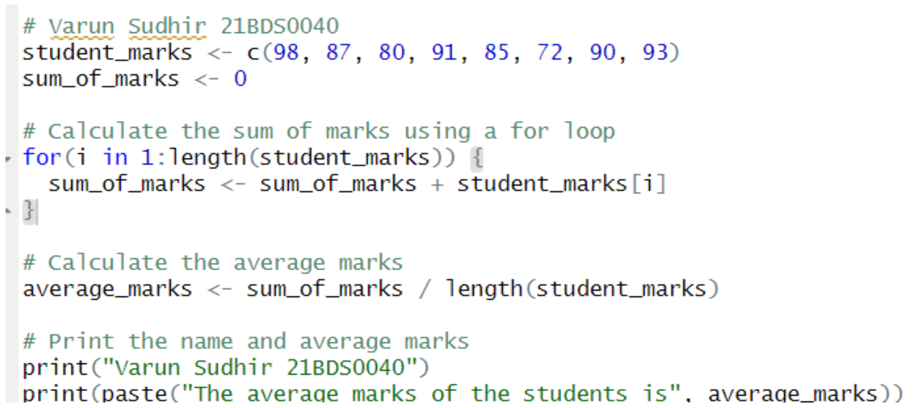
Reg No: 21BDS0040

Exploratory Data Analysis Assignment – 1

1) Given an input vector of the marks of students in a class, find the average marks of the class

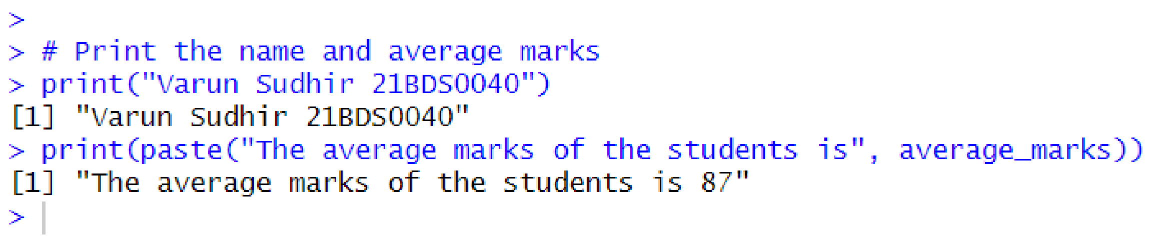
Code:

```r
# Varun Sudhir 21BDS0040
student_marks <- c(98, 87, 80, 91, 85, 72, 90, 93)
sum_of_marks <- 0

# Calculate the sum of marks using a for loop
for(i in 1:length(student_marks)) {
  sum_of_marks <- sum_of_marks + student_marks[i]
}

# Calculate the average marks
average_marks <- sum_of_marks / length(student_marks)

# Print the name and average marks
print("Varun Sudhir 21BDS0040")
print(paste("The average marks of the students is", average_marks))
```



```r
# Varun Sudhir 21BDS0040
student_marks <- c(98, 87, 80, 91, 85, 72, 90, 93)
sum_of_marks <- 0

# Calculate the sum of marks using a for loop
for(i in 1:length(student_marks)) {
  sum_of_marks <- sum_of_marks + student_marks[i]
}

# Calculate the average marks
average_marks <- sum_of_marks / length(student_marks)

# Print the name and average marks
print("Varun Sudhir 21BDS0040")
print(paste("The average marks of the students is", average_marks))
```

Output:

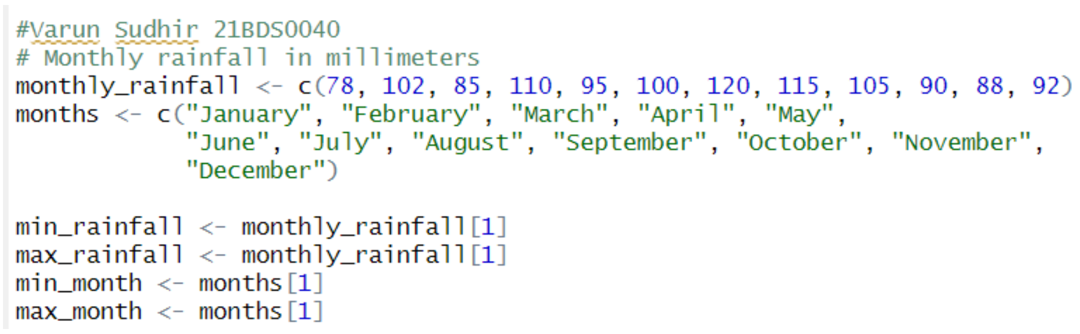


```
>
> # Print the name and average marks
> print("Varun Sudhir 21BDS0040")
[1] "Varun Sudhir 21BDS0040"
> print(paste("The average marks of the students is", average_marks))
[1] "The average marks of the students is 87"
> 
```

2) Given an input vectors of the monthly rainfalls over a year, calculate the minimum and maximum rainfall month in the year

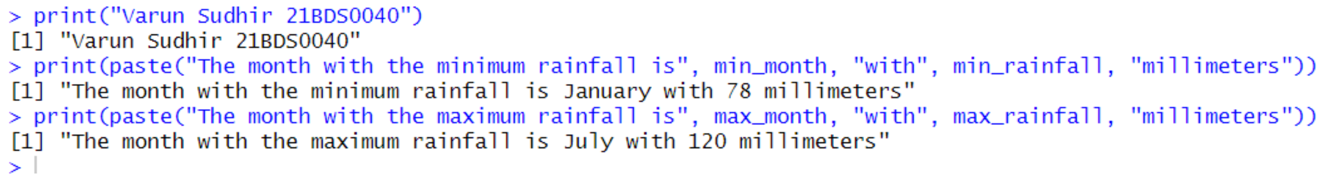
Code:

```
#Varun Sudhir 21BDS0040
#Monthly rainfall in millimeters
monthly_rainfall <- c(78, 102, 85, 110, 95, 100, 120, 115, 105, 90, 88, 
92)
months <- c("January", "February", "March", "April", "May",
            "June", "July", "August", "September", "October",
      "November", "December")

min_rainfall <- monthly_rainfall[1]
max_rainfall <- monthly_rainfall[1]
min_month <- months[1]
max_month <- months[1]

# Iterate through the monthly rainfall data to find the min and max 
rainfall
for (i in 2:length(monthly_rainfall))
{
  if (monthly_rainfall[i] < min_rainfall)
  {
    min_rainfall <- monthly_rainfall[i]
    min_month <- months[i]
  }
  if (monthly_rainfall[i] > max_rainfall)
  {
    max_rainfall <- monthly_rainfall[i]
    max_month <- months[i]
  }
}
print(paste("The month with the minimum rainfall is", min_month, 
"with", min_rainfall, "millimeters"))
print(paste("The month with the maximum rainfall is", max_month, 
"with", max_rainfall, "millimeters"))
```



```
#Varun Sudhir 21BDS0040
# Monthly rainfall in millimeters
monthly_rainfall <- c(78, 102, 85, 110, 95, 100, 120, 115, 105, 90, 88, 92)
months <- c("January", "February", "March", "April", "May",
            "June", "July", "August", "September", "October", "November",
            "December")

min_rainfall <- monthly_rainfall[1]
max_rainfall <- monthly_rainfall[1]
min_month <- months[1]
max_month <- months[1]
```

Output:

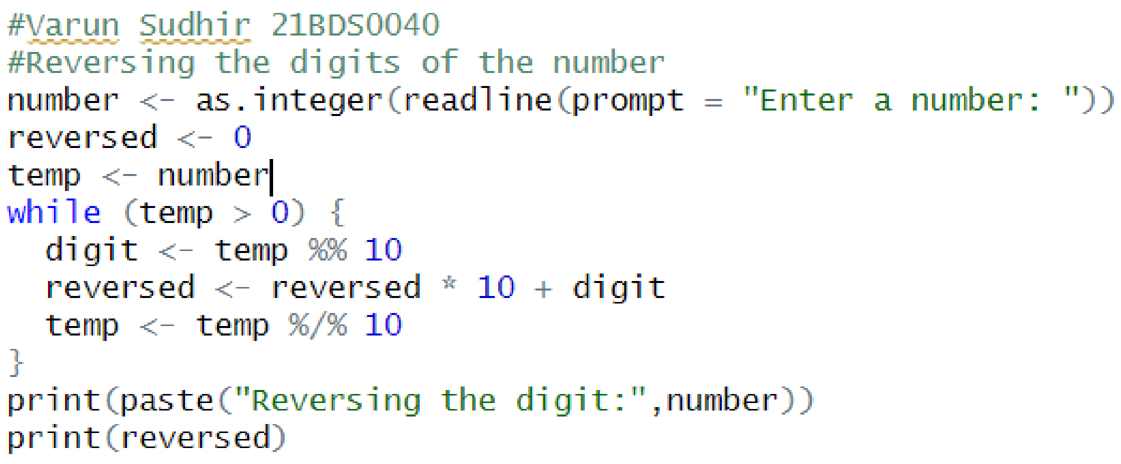


```
> print("Varun Sudhir 21BDS0040")
[1] "Varun Sudhir 21BDS0040"
> print(paste("The month with the minimum rainfall is", min_month, "with", min_rainfall, "millimeters"))
[1] "The month with the minimum rainfall is January with 78 millimeters"
> print(paste("The month with the maximum rainfall is", max_month, "with", max_rainfall, "millimeters"))
[1] "The month with the maximum rainfall is July with 120 millimeters"
> 
```

3) Given a number, reverse the number and display it

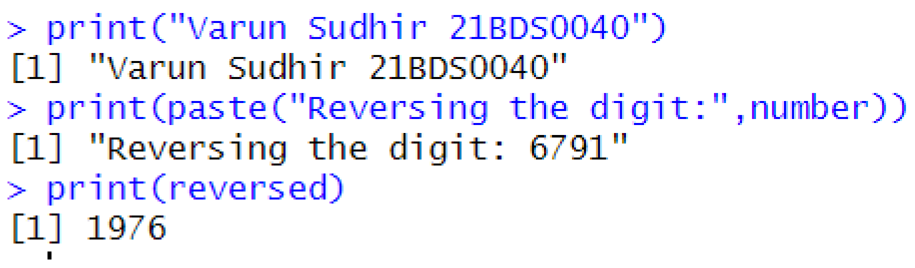
Code:

```
#Varun Sudhir 21BDS0040
#Reversing the digits of the number
number <- as.integer(readline(prompt = "Enter a number: "))
reversed <- 0
temp <- number
while (temp > 0) {
  digit <- temp %% 10
  reversed <- reversed * 10 + digit
  temp <- temp %/% 10
}
print(paste("Reversing the digit:",number))
print(reversed)
```



```
#Varun Sudhir 21BDS0040
#Reversing the digits of the number
number <- as.integer(readline(prompt = "Enter a number: "))
reversed <- 0
temp <- number
while (temp > 0) {
  digit <- temp %% 10
  reversed <- reversed * 10 + digit
  temp <- temp %/% 10
}
print(paste("Reversing the digit:",number))
print(reversed)
```

Output:

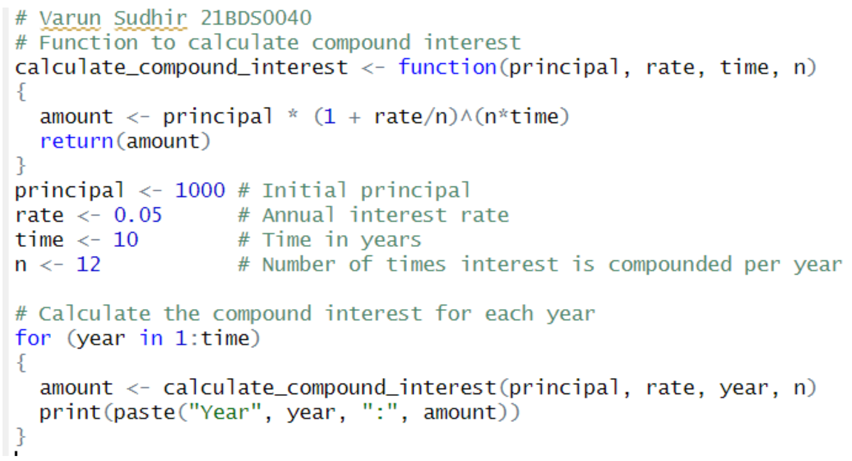


```
> print("Varun Sudhir 21BDS0040")
[1] "Varun Sudhir 21BDS0040"
> print(paste("Reversing the digit:",number))
[1] "Reversing the digit: 6791"
> print(reversed)
[1] 1976
```

4) Calculating compound interest over time using a for-loop

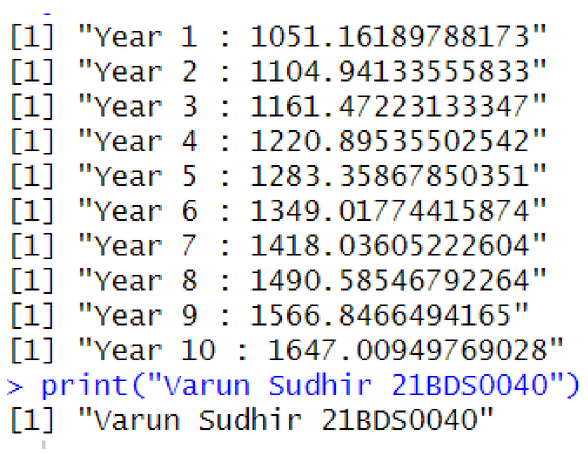
Code:

```r
# Varun Sudhir 21BDS0040
# Function to calculate compound interest
calculate_compound_interest <- function(principal, rate, time, n)
{
  amount <- principal * (1 + rate/n)^(n*time)
  return(amount)
}
principal <- 1000 # Initial principal
rate <- 0.05       # Annual interest rate
time <- 10         # Time in years
n <- 12            # Number of times interest is compounded per year
# Calculate the compound interest for each year
for (year in 1:time)
{
  amount <- calculate_compound_interest(principal, rate, year, n)
  print(paste("Year", year, ":", amount))
}
```



```r
# Varun Sudhir 21BDS0040
# Function to calculate compound interest
calculate_compound_interest <- function(principal, rate, time, n)
{
  amount <- principal * (1 + rate/n)^(n*time)
  return(amount)
}
principal <- 1000 # Initial principal
rate <- 0.05       # Annual interest rate
time <- 10         # Time in years
n <- 12            # Number of times interest is compounded per year

# Calculate the compound interest for each year
for (year in 1:time)
{
  amount <- calculate_compound_interest(principal, rate, year, n)
  print(paste("Year", year, ":", amount))
}
```

Output:

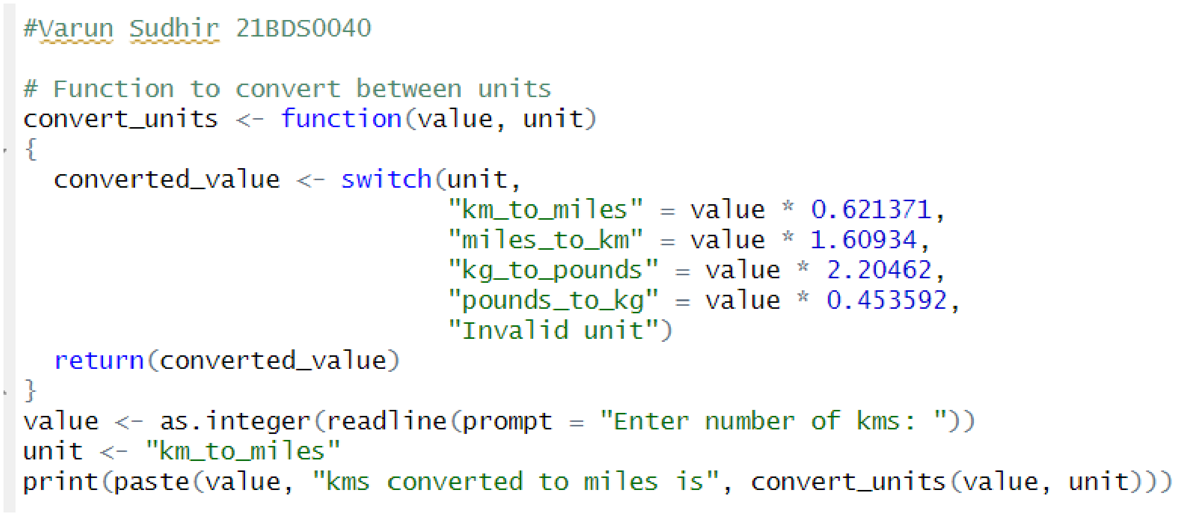


```
[1] "Year 1 : 1051.16189788173"
[1] "Year 2 : 1104.94133555833"
[1] "Year 3 : 1161.47223133347"
[1] "Year 4 : 1220.89535502542"
[1] "Year 5 : 1283.35867850351"
[1] "Year 6 : 1349.01774415874"
[1] "Year 7 : 1418.03605222604"
[1] "Year 8 : 1490.58546792264"
[1] "Year 9 : 1566.8466494165"
[1] "Year 10 : 1647.00949769028"
> print("Varun Sudhir 21BDS0040")
[1] "Varun Sudhir 21BDS0040"
```

5) To convert units of measurements with the help of switch statement

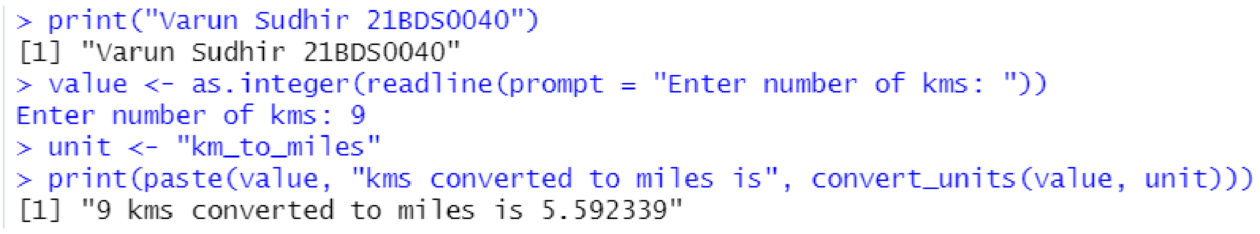
Code:

```
#Varun Sudhir 21BDS0040
# Function to convert between units
convert_units <- function(value, unit)
{
  converted_value <- switch(unit,
                            "km_to_miles" = value * 0.621371,
                            "miles_to_km" = value * 1.60934,
                            "kg_to_pounds" = value * 2.20462,
                            "pounds_to_kg" = value * 0.453592,
                            "Invalid unit")
  return(converted_value)
}
value <- as.integer(readline(prompt = "Enter number of kms: "))
unit <- "km_to_miles"
print(paste(value, "kms converted to miles is", convert_units(value,
unit)))
```



```
#Varun Sudhir 21BDS0040

# Function to convert between units
convert_units <- function(value, unit)
{
  converted_value <- switch(unit,
                            "km_to_miles" = value * 0.621371,
                            "miles_to_km" = value * 1.60934,
                            "kg_to_pounds" = value * 2.20462,
                            "pounds_to_kg" = value * 0.453592,
                            "Invalid unit")
  return(converted_value)
}
value <- as.integer(readline(prompt = "Enter number of kms: "))
unit <- "km_to_miles"
print(paste(value, "kms converted to miles is", convert_units(value, unit)))
```

Output:

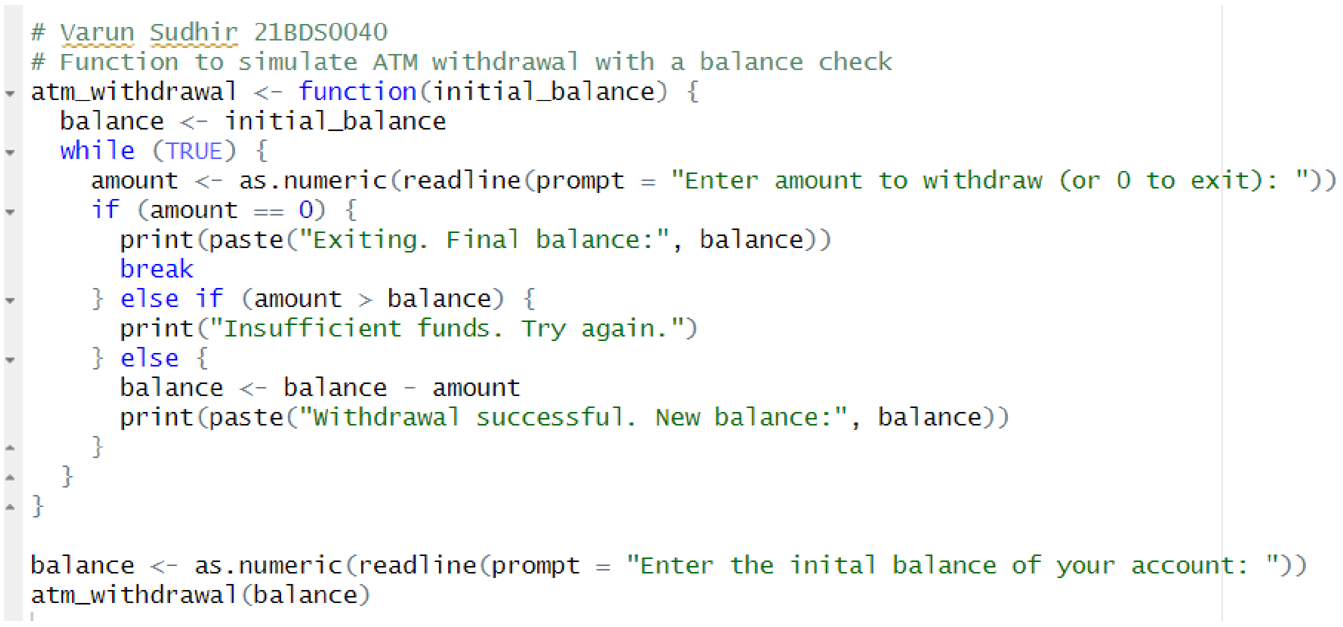


```
> print("Varun Sudhir 21BDS0040")
[1] "Varun Sudhir 21BDS0040"
> value <- as.integer(readline(prompt = "Enter number of kms: "))
Enter number of kms: 9
> unit <- "km_to_miles"
> print(paste(value, "kms converted to miles is", convert_units(value, unit)))
[1] "9 kms converted to miles is 5.592339"
```

6) Simulating a sample ATM withdrawal , money can be withdrawn from the initial balance in your account

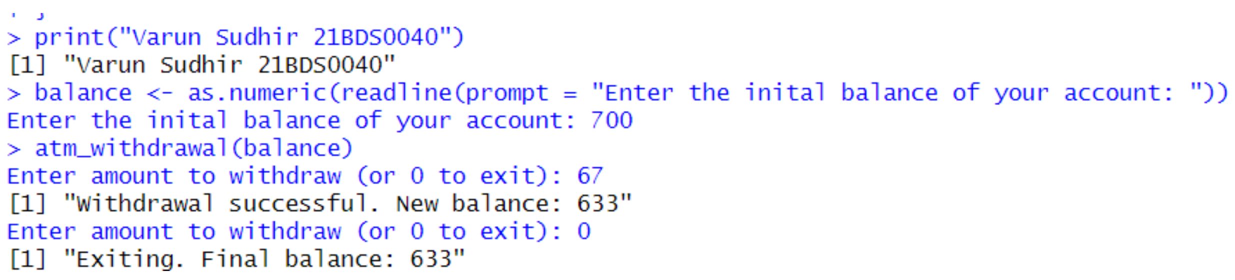
Code:

```
# Varun Sudhir 21BDS0040
# Function to simulate ATM withdrawal with a balance check

atm_withdrawal <- function(initial_balance)
{
  balance <- initial_balance
  while (TRUE) {
    amount <- as.numeric(readline(prompt = "Enter amount to withdraw
(or 0 to exit): "))
    if (amount == 0) {
      print(paste("Exiting. Final balance:", balance))
      break
    } else if (amount > balance) {
      print("Insufficient funds. Try again.")
    } else {
      balance <- balance - amount
      print(paste("Withdrawal successful. New balance:", balance))
    }
  }
}
balance <- as.numeric(readline(prompt = "Enter the inital balance of
your account: "))
atm_withdrawal(balance)
```



```
# Varun Sudhir 21BDS0040
# Function to simulate ATM withdrawal with a balance check
atm_withdrawal <- function(initial_balance) {
  balance <- initial_balance
  while (TRUE) {
    amount <- as.numeric(readline(prompt = "Enter amount to withdraw (or 0 to exit): "))
    if (amount == 0) {
      print(paste("Exiting. Final balance:", balance))
      break
    } else if (amount > balance) {
      print("Insufficient funds. Try again.")
    } else {
      balance <- balance - amount
      print(paste("Withdrawal successful. New balance:", balance))
    }
  }
}

balance <- as.numeric(readline(prompt = "Enter the inital balance of your account: "))
atm_withdrawal(balance)
```

Output:

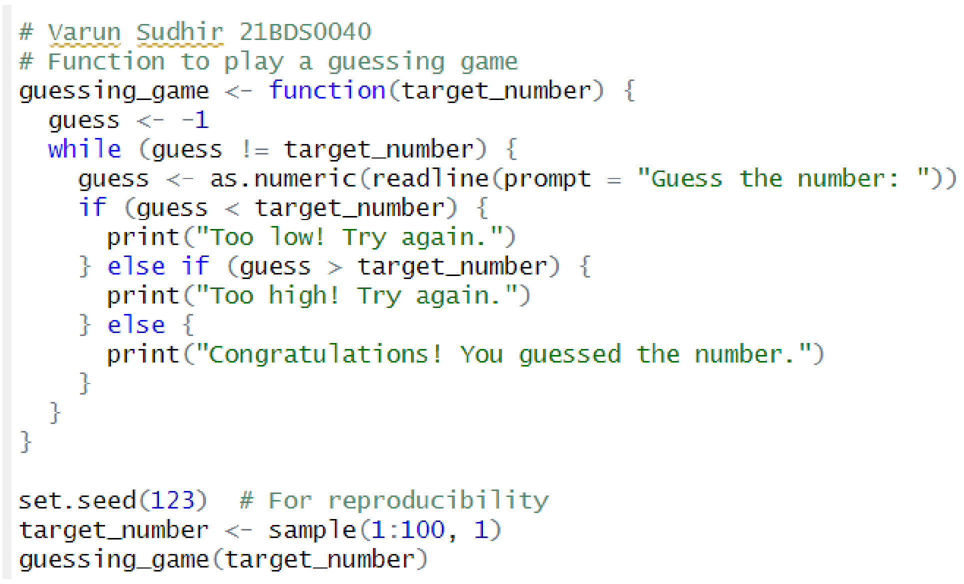


```
> print("Varun Sudhir 21BDS0040")
[1] "Varun Sudhir 21BDS0040"
> balance <- as.numeric(readline(prompt = "Enter the inital balance of your account: "))
Enter the inital balance of your account: 700
> atm_withdrawal(balance)
Enter amount to withdraw (or 0 to exit): 67
[1] "Withdrawal successful. New balance: 633"
Enter amount to withdraw (or 0 to exit): 0
[1] "Exiting. Final balance: 633"
```

7) Simulating a Guessing Game with the help of user input using a while loop

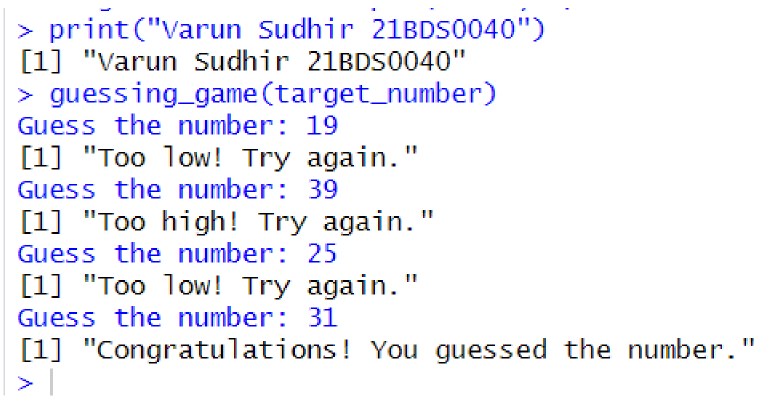
Code:

```
# Varun Sudhir 21BDS0040
# Function to play a guessing game
guessing_game <- function(target_number) {
  guess <- -1
  while (guess != target_number) {
    guess <- as.numeric(readline(prompt = "Guess the number: "))
    if (guess < target_number) {
      print("Too low! Try again.")
    } else if (guess > target_number) {
      print("Too high! Try again.")
    } else {
      print("Congratulations! You guessed the number.")
    }
  }
}
set.seed(123)  # For reproducibility
target_number <- sample(1:100, 1)
print(“Varun Sudhir 21BDS0040”)
guessing_game(target_number)
```



```
# Varun Sudhir 21BDS0040
# Function to play a guessing game
guessing_game <- function(target_number) {
  guess <- -1
  while (guess != target_number) {
    guess <- as.numeric(readline(prompt = "Guess the number: "))
    if (guess < target_number) {
      print("Too low! Try again.")
    } else if (guess > target_number) {
      print("Too high! Try again.")
    } else {
      print("Congratulations! You guessed the number.")
    }
  }
}

set.seed(123)  # For reproducibility
target_number <- sample(1:100, 1)
guessing_game(target_number)
```

Output:



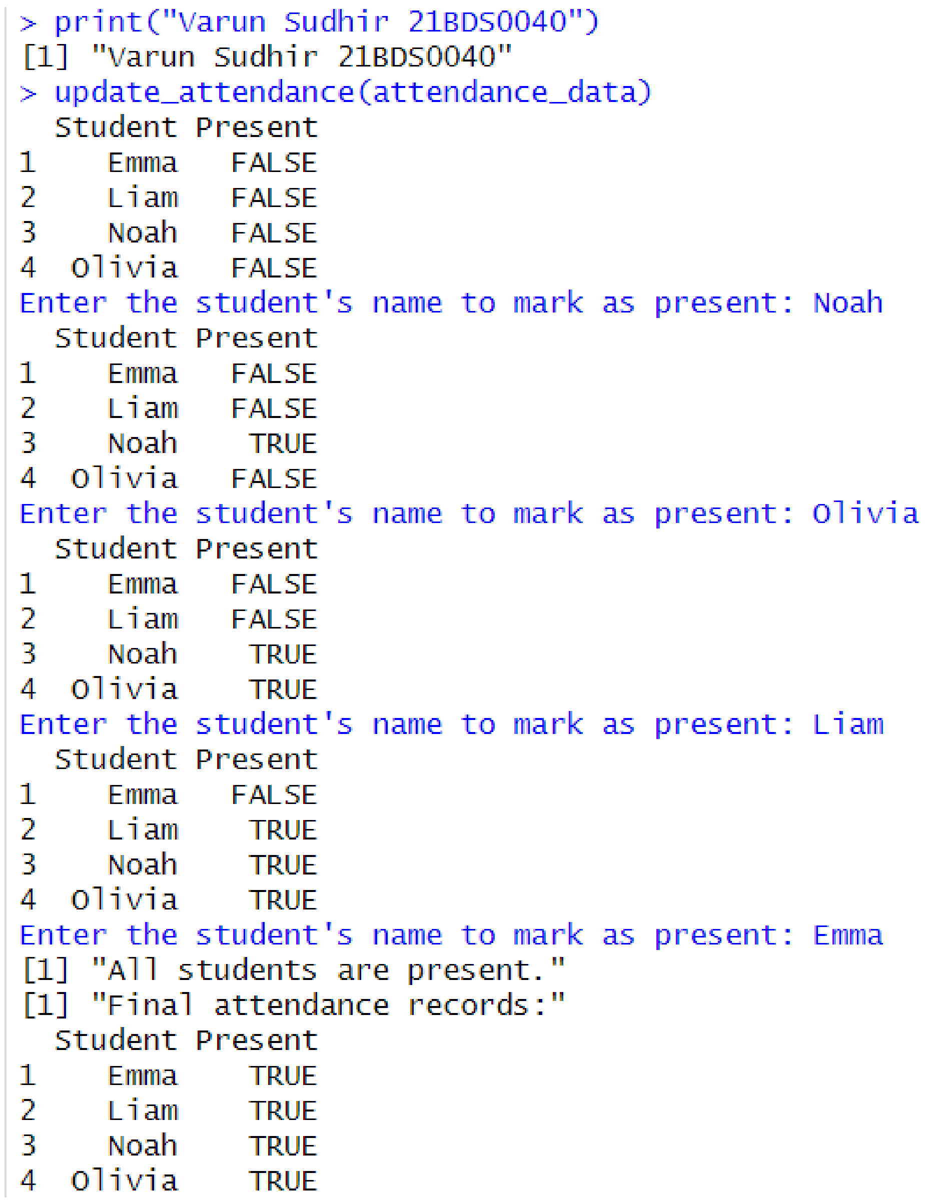
```
> print("Varun Sudhir 21BDS0040")
[1] "Varun Sudhir 21BDS0040"
> guessing_game(target_number)
Guess the number: 19
[1] "Too low! Try again."
Guess the number: 39
[1] "Too high! Try again."
Guess the number: 25
[1] "Too low! Try again."
Guess the number: 31
[1] "Congratulations! You guessed the number."
> 
```

8) Building a Student Monitoring system that will track the attendance of the students and mark all the students as Present

Code:

```
# Varun Sudhir 21BDS0040
# Function to update attendance records

update_attendance <- function(data) {
  while (any(!data$Present)) {
    print(data)

    # Prompt user for student name and attendance status
    student <- readline(prompt = "Enter the student's name to mark as 
present: ")

    # Update the attendance if the student is found
    if (student %in% data$Student) {
      data <- data %>%
        mutate(Present = ifelse(Student == student, TRUE, Present))
    } else {
      print("Student not found. Please enter a valid name.")
    }

    if (all(data$Present)) {
      print("All students are present.")
      break
    }
  }
  print("Final attendance records:")
  print(data)
}

attendance_data <- data.frame(
  Student = c("Emma", "Liam", "Noah", "Olivia"),
  Present = c(FALSE, FALSE, FALSE, FALSE),
  stringsAsFactors = FALSE
)
print(“Varun Sudhir 21BDS0040”)
update_attendance(attendance_data)
```

Output:



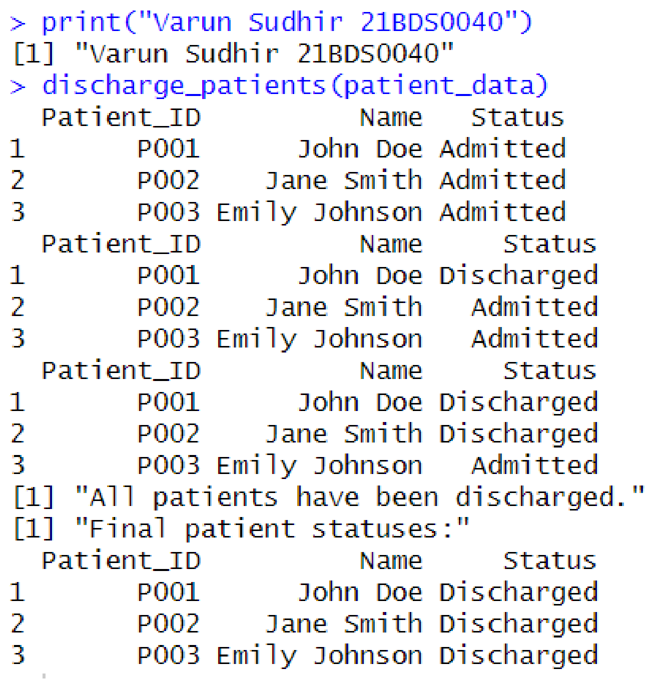
```
> print("Varun Sudhir 21BDS0040")
[1] "Varun Sudhir 21BDS0040"
> update_attendance(attendance_data)
  Student Present
1    Emma   FALSE
2    Liam   FALSE
3    Noah   FALSE
4  Olivia   FALSE
Enter the student's name to mark as present: Noah
  Student Present
1    Emma   FALSE
2    Liam   FALSE
3    Noah    TRUE
4  Olivia   FALSE
Enter the student's name to mark as present: Olivia
  Student Present
1    Emma   FALSE
2    Liam   FALSE
3    Noah    TRUE
4  Olivia    TRUE
Enter the student's name to mark as present: Liam
  Student Present
1    Emma   FALSE
2    Liam    TRUE
3    Noah    TRUE
4  Olivia    TRUE
Enter the student's name to mark as present: Emma
[1] "All students are present."
[1] "Final attendance records:"
  Student Present
1    Emma    TRUE
2    Liam    TRUE
3    Noah    TRUE
4  Olivia    TRUE
```

9) To discharge the patients of a hospital one-by-one , until all the patients of the hospital are discharged

Code:

```r
#Varun Sudhir 21BDS0040
discharge_patients <- function(data) {
  index <- 1
  while (index <= nrow(data) && any(data$Status == "Admitted")) {
    print(data)
    data$Status[index] <- "Discharged"
    index <- index + 1
    # Check if all patients are discharged
    if (all(data$Status == "Discharged")) {
      print("All patients have been discharged.")
    }
  }

  print("Final patient statuses:")
  print(data)
}
patient_data <- data.frame(
  Patient_ID = c("P001", "P002", "P003"),
  Name = c("John Doe", "Jane Smith", "Emily Johnson"),
  Status = c("Admitted", "Admitted", "Admitted"),
  stringsAsFactors = FALSE
)
print(“Varun Sudhir 21BDS0040 ”)
discharge_patients(patient_data)
```

Output:



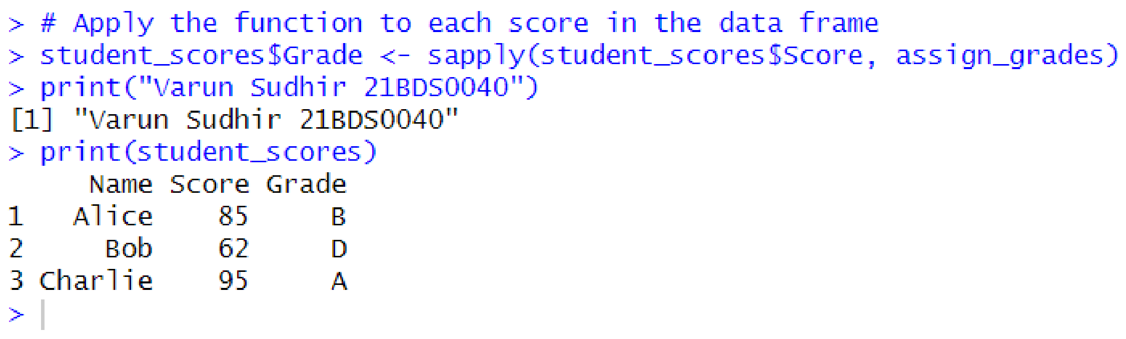
```
> print("Varun Sudhir 21BDS0040")
[1] "Varun Sudhir 21BDS0040"
> discharge_patients(patient_data)
  Patient_ID          Name   Status
1       P001      John Doe Admitted
2       P002    Jane Smith Admitted
3       P003 Emily Johnson Admitted
  Patient_ID          Name     Status
1       P001      John Doe Discharged
2       P002    Jane Smith   Admitted
3       P003 Emily Johnson   Admitted
  Patient_ID          Name     Status
1       P001      John Doe Discharged
2       P002    Jane Smith Discharged
3       P003 Emily Johnson   Admitted
[1] "All patients have been discharged."
[1] "Final patient statuses:"
  Patient_ID          Name     Status
1       P001      John Doe Discharged
2       P002    Jane Smith Discharged
3       P003 Emily Johnson Discharged
```

10) To assign a grade to a student based on their marks using if-else statements

Code:

```
#Varun Sudhir 21BDS0040
student_scores <- data.frame(
  Name = c("Alice", "Bob", "Charlie"),
  Score = c(85, 62, 95),
  stringsAsFactors = FALSE
)

# Function to assign grades based on score
assign_grades <- function(score) {
  if (score >= 90) {
    return("A")
  } else if (score >= 80) {
    return("B")
  } else if (score >= 70) {
    return("C")
  } else if (score >= 60) {
    return("D")
  } else {
    return("F")
  }
}

# Apply the function to each score in the data frame
student_scores$Grade <- sapply(student_scores$Score, assign_grades)

print("Varun Sudhir 21BDS0040")
print(student_scores)
```

Output:



```
> # Apply the function to each score in the data frame
> student_scores$Grade <- sapply(student_scores$Score, assign_grades)
> print("Varun Sudhir 21BDS0040")
[1] "Varun Sudhir 21BDS0040"
> print(student_scores)
     Name Score Grade
1   Alice    85     B
2     Bob    62     D
3 Charlie    95     A
> 
```