EXPT NO : 08	ITERATION AND CONDITIONAL STATEMENTS
DATE : 19.03.2025	

AIM:

To Implement iteration and conditional statements using r programming.

1. Conditional Statements:

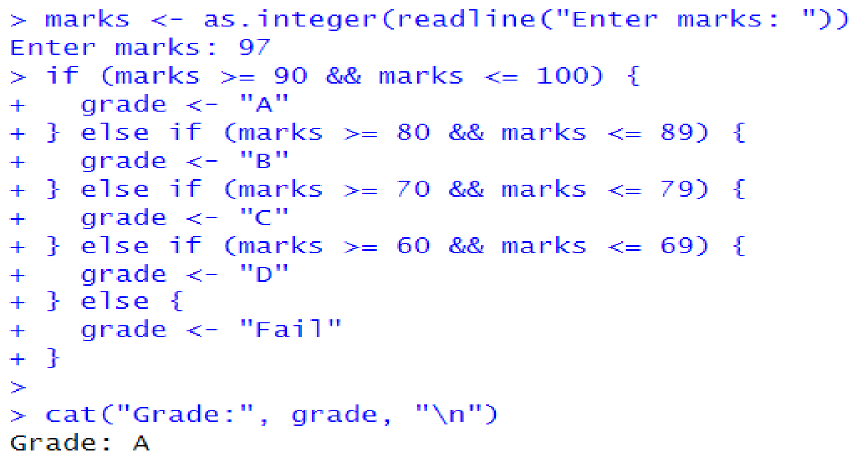
• Write an R program to accept marks and print the corresponding grade based on thefollowing criteria:

o 90-100 → A o 80-89 → B o 70-79 → C o 60-69 → D o Below 60 → Fail

CODE:

```
marks <- as.integer(readline("Enter marks: "))

if (marks >= 90 && marks <= 100) {grade <- "A"}

 else if (marks >= 80 && marks <= 89)  {grade <- "B"}

 else if (marks >= 70 && marks <= 79) {grade <- "C"}

 else if (marks >= 60 && marks <= 69) {grade <- "D"}

 else {grade <- "Fail" }

cat("Grade:", grade, "\n")
```

OUTPUT:



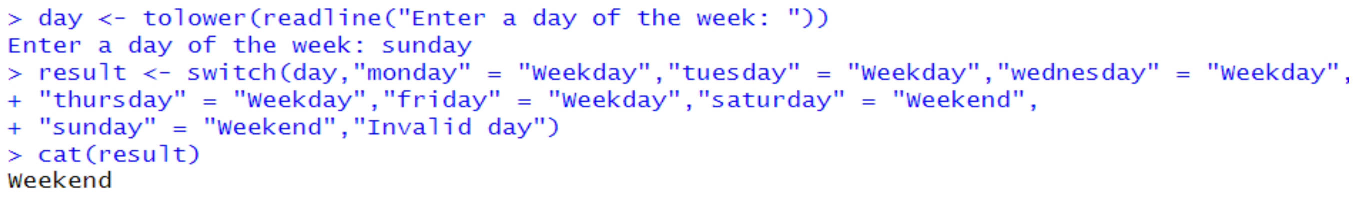
```
> marks <- as.integer(readline("Enter marks: "))
Enter marks: 97
> if (marks >= 90 && marks <= 100) {
+   grade <- "A"
+ } else if (marks >= 80 && marks <= 89) {
+   grade <- "B"
+ } else if (marks >= 70 && marks <= 79) {
+   grade <- "C"
+ } else if (marks >= 60 && marks <= 69) {
+   grade <- "D"
+ } else {
+   grade <- "Fail"
+ }
>
> cat("Grade:", grade, "\n")
Grade: A
```

• Write an R program to take a character input representing a day of the week and print whether it is a weekday or weekend using a switch statement.

CODE:

```
day <- tolower(readline("Enter a day of the week: "))

result <- switch(day,"monday" = "Weekday","tuesday" = "Weekday","wednesday" = "Weekday",

"thursday" = "Weekday","friday" = "Weekday","saturday" = "Weekend",

"sunday" = "Weekend","Invalid day")
```

OUTPUT:



```
> day <- tolower(readline("Enter a day of the week: "))
Enter a day of the week: sunday
> result <- switch(day,"monday" = "Weekday","tuesday" = "Weekday","wednesday" = "Weekday",
+ "thursday" = "Weekday","friday" = "Weekday","saturday" = "Weekend",
+ "sunday" = "Weekend","Invalid day")
> cat(result)
Weekend
```

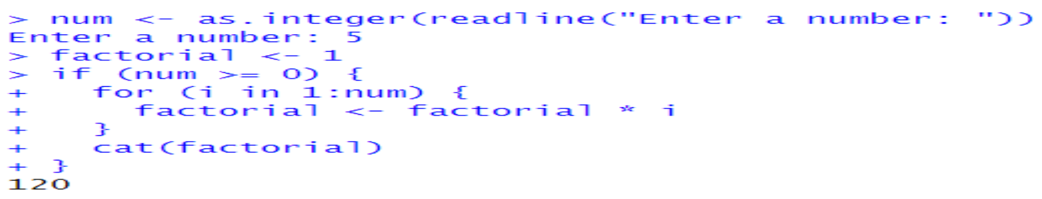
2. Iteration Statements:

• Write an R program to compute the factorial of a given number using a for loop.

CODE:

```
num <- as.integer(readline("Enter a number: "))

factorial <- 1

if (num >= 0) {

  for (i in 1:num) {

    factorial <- factorial * I }

  cat(factorial) }
```

OUTPUT:



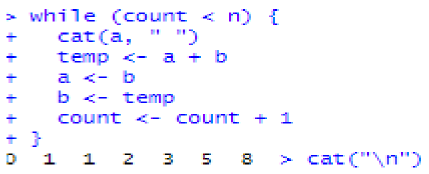
```
> num <- as.integer(readline("Enter a number: "))
Enter a number: 5
> factorial <- 1
> if (num >= 0) {
+     for (i in 1:num) {
+         factorial <- factorial * i
+     }
+     cat(factorial)
+ }
120
```

• Write an R program to generate the first N terms of the Fibonacci sequence using a while loop.

CODE:

```
n <- as.integer(readline("Enter the number of terms: "))

a <- 0

b <- 1

count <- 0

cat("Fibonacci Sequence: ")while (count < n) { cat(a, " ")

  temp <- a + b

  a <- b

  b <- temp

  count <- count + 1}
```

OUTPUT:



```
> while (count < n) {
+   cat(a, " ")
+   temp <- a + b
+   a <- b
+   b <- temp
+   count <- count + 1
+ }
0  1  1  2  3  5  8  > cat("\n")
```

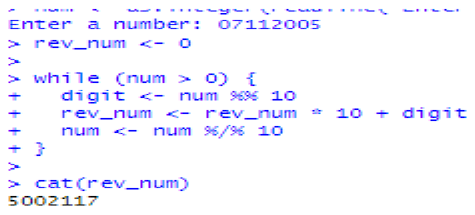
3. Nested Loops and Conditions:

• Write an R program to reverse a given number using a loop.

CODE:

```
num <- as.integer(readline("Enter a number: "))

rev_num <- 0

while (num > 0) {

  digit <- num %% 10

  rev_num <- rev_num * 10 + digit

  num <- num %/% 10

}
```

OUTPUT:



```
Enter a number: 07112005
> rev_num <- 0
>
> while (num > 0) {
+   digit <- num %% 10
+   rev_num <- rev_num * 10 + digit
+   num <- num %/% 10
+ }
>
> cat(rev_num)
5002117
```

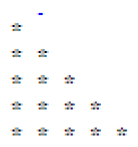
• Write an R program to print the following pattern using nested loops:

```
*

* *

* * *

* * * *

* * * * *
```

CODE:

```
rows <- 5

for (i in 1:rows) {

  for (j in 1:i) {
    cat("* " }
  cat("\n")  }
```

OUTPUT:



```
*
* *
* * *
* * * *
* * * * *
```

4. Problem-Solving:

• Write an R program to simulate a simple ATM system with the following functionalities.

o Allow the user to enter an initial account balance.

o Provide options to deposit money, withdraw money, or check the current balance using loops.

o Ensure the program runs continuously until the user chooses to exit.

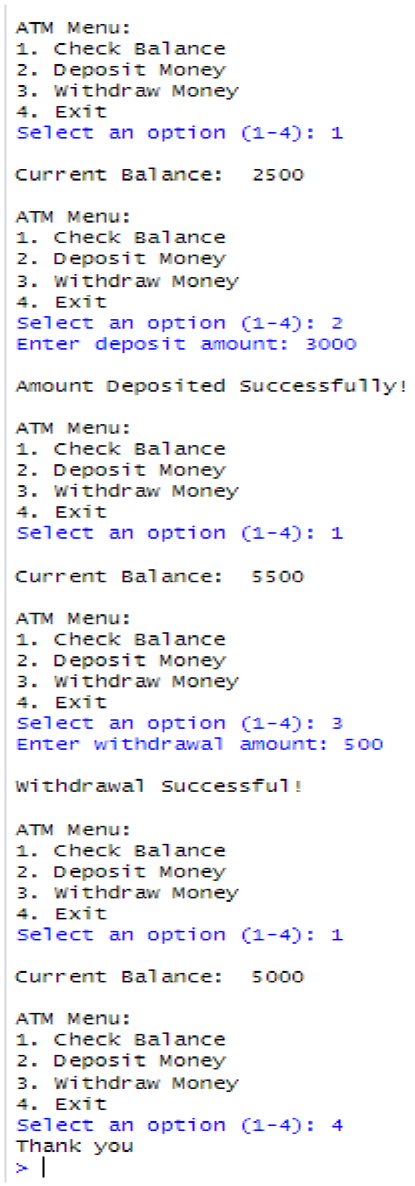
o Implement necessary validations, such as ensuring the withdrawal amount does not exceed the available balance.

CODE:

```
account_balance <- as.numeric(readline("Enter initial account balance: "))
display_menu <- function() {
  cat("\nATM Menu:\n")
  cat("1. Check Balance\n")
  cat("2. Deposit Money\n")
  cat("3. Withdraw Money\n")
  cat("4. Exit\n")
}
repeat {
  display_menu()
  choice <- as.integer(readline("Select an option (1-4): "))

  if (choice == 1) {
    cat("\nCurrent Balance: ", account_balance, "\n")
  } else if (choice == 2) {
   deposit <- as.numeric(readline("Enter deposit amount: "))
```

```
    if (deposit > 0) {
      account_balance <- account_balance + deposit
      cat("\nAmount Deposited Successfully!\n")
    } else {
      cat("\nInvalid Deposit Amount!\n")
    }
  } else if (choice == 3) {
    withdraw <- as.numeric(readline("Enter withdrawal amount: "))
    if (withdraw > 0 && withdraw <= account_balance) {
      account_balance <- account_balance - withdraw
      cat("\nWithdrawal Successful!\n")
    } else {
      cat("\nInvalid or Insufficient Balance!\n")
    }
  } else if (choice == 4) {
    cat("Thank you ")
    break
  } else {
    cat("Invalid Choice")
  }
}
```

OUTPUT:



```
ATM Menu:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Select an option (1-4): 1

Current Balance:  2500

ATM Menu:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Select an option (1-4): 2
Enter deposit amount: 3000

Amount Deposited Successfully!

ATM Menu:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Select an option (1-4): 1

Current Balance:  5500

ATM Menu:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Select an option (1-4): 3
Enter withdrawal amount: 500

Withdrawal Successful!

ATM Menu:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Select an option (1-4): 1

Current Balance:  5000

ATM Menu:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Select an option (1-4): 4
Thank you
> |
```

RESULT:

Thus , the iteration and conditional statements is implemented and output is verified successfully.