**Aim:** Understanding Looping and Loop Control Statements using R

**IDE:** R Studio

**Theory:**

**Looping statements** are expressions used to control the execution and flow of the program based on the conditions provided in the statements. These structures are used to make a decision after assessing the variable.

In [R programming](https://www.geeksforgeeks.org/introduction-to-r-programming-language/), there are 7 types of loop control statements as follows:

* **if condition**
* **if-else condition**
* **for loop**
* **nested loops**
* **while loop**
* **break statement**
* **next statement**

#### **if condition**

This control structure checks the expression provided in parenthesis is true or not. If true, the execution of the statements in braces {} continues.

if(expression){

statements

.... ....

}

#### **if-else condition**

It is similar to **if** condition but when the test expression in if condition fails, then statements in **else** condition are executed.

if(expression){

statements

.... ....

}

else{

statements

.... ....

}

#### **for loop**

It is a type of loop or sequence of statements executed repeatedly until exit condition is reached.

for(value in vector){

statements

.... ....

}

#### **Nested loops**

Nested loops are similar to simple loops. Nested means loops inside loop. Moreover, nested loops are used to manipulate the matrix.

#### while loop

while loop is another kind of loop iterated until a condition is satisfied. The testing expression is checked first before executing the body of loop.

while(expression){

statement

.... ....

}

#### **break statement**

**break**statement can be used in any type of loop to exit from the loop.

if(expression) {

break

}

#### **next statement**

**next**statement is used to skip the current iteration without executing the further statements and continues the next iteration cycle without terminating the loop.

**Programs:**

Write R script that demonstrates the functionality of all the loop control statements:

1. Print the total of first 100 even numbers
2. Print 0 1 1 2 3 5 8 13 21 34 55 … (Fibonanci series)
3. Generate the value of n! factorial of a number n
4. Except one no from user and find if it is Armstrong or not.
5. Accept a number from user and display Reverse of it.
6. Accept a number of length n (n>3) from user and display its sum of digits.
7. Print pattern like follow.

12345

12345

12345

12345

12345

1. Print pattern like follow.

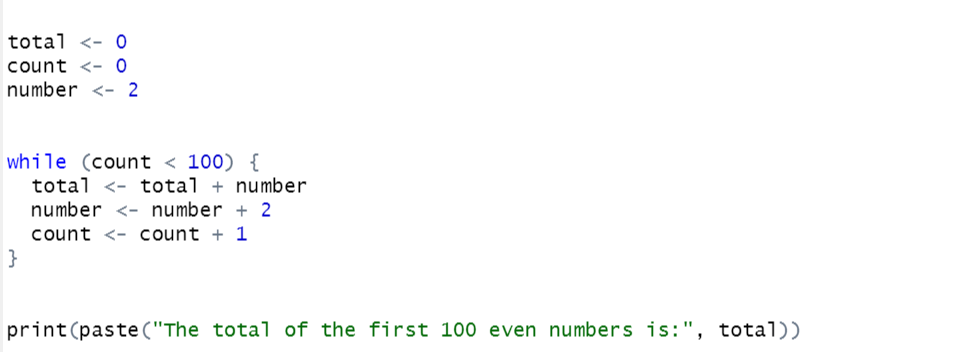
A A A A A

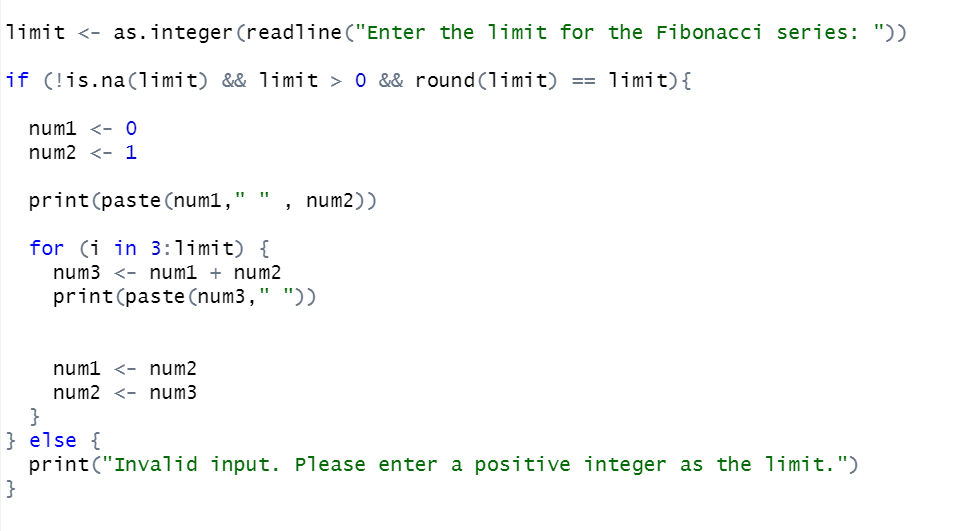
B B B B B

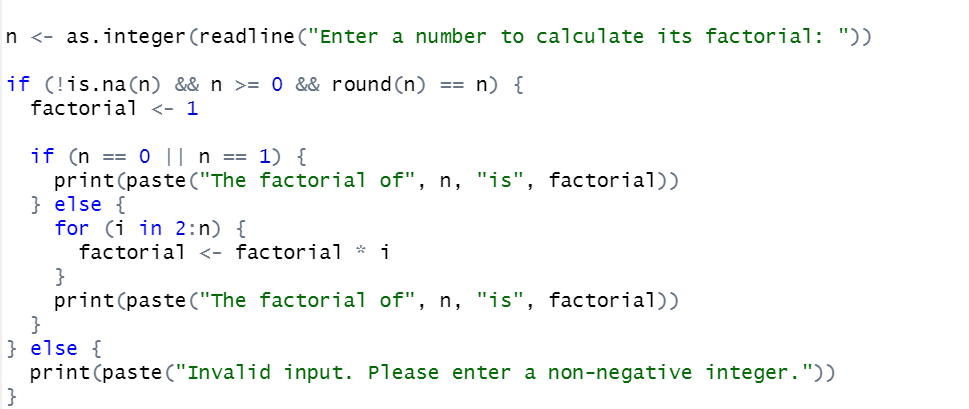
C C C C C

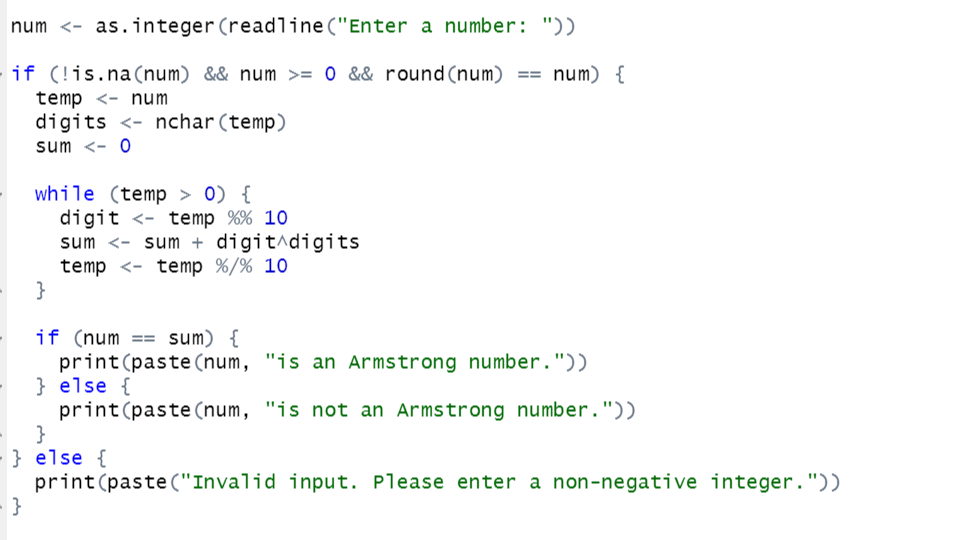
D D D D D

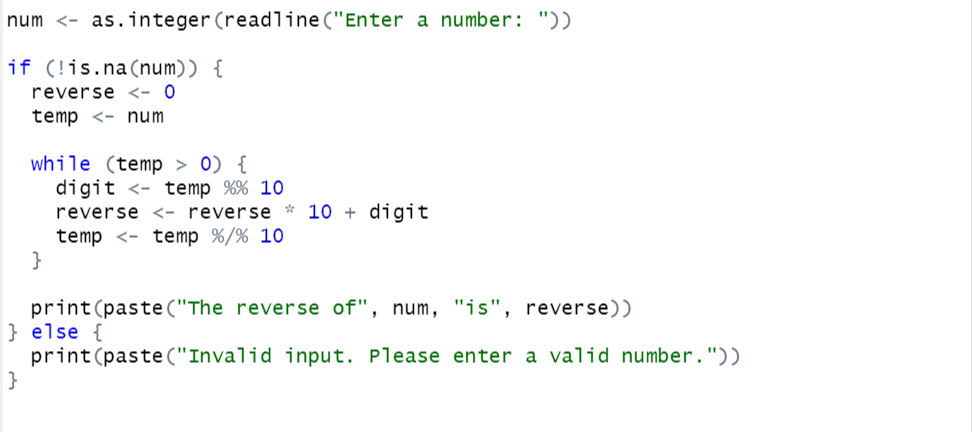
E E E E E

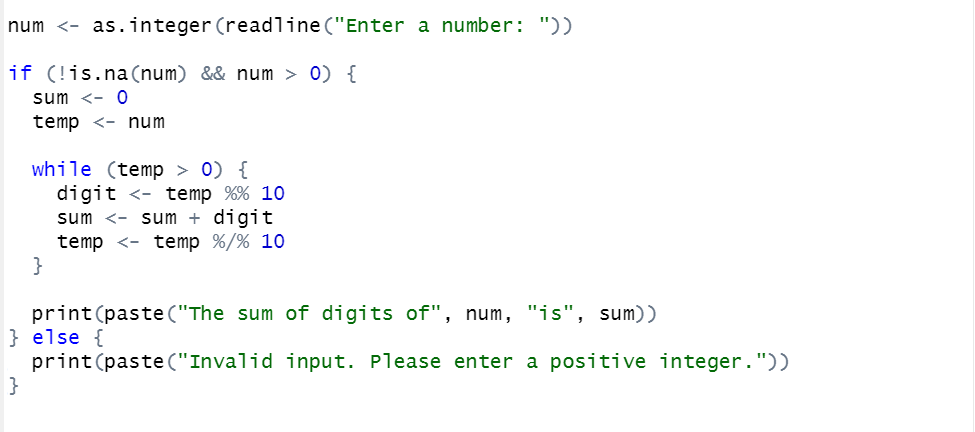


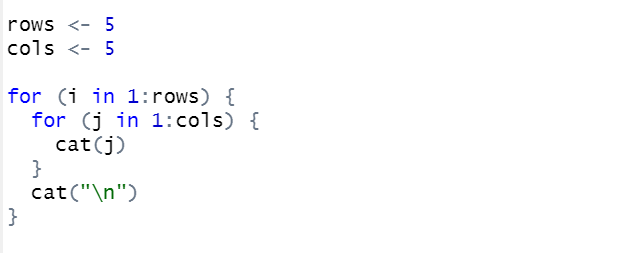
****

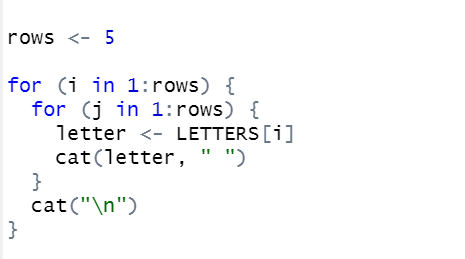
****

****

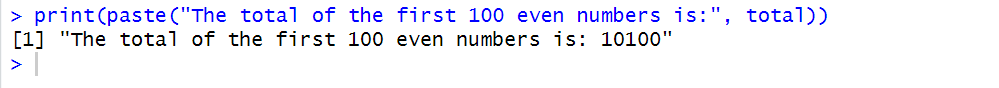
****

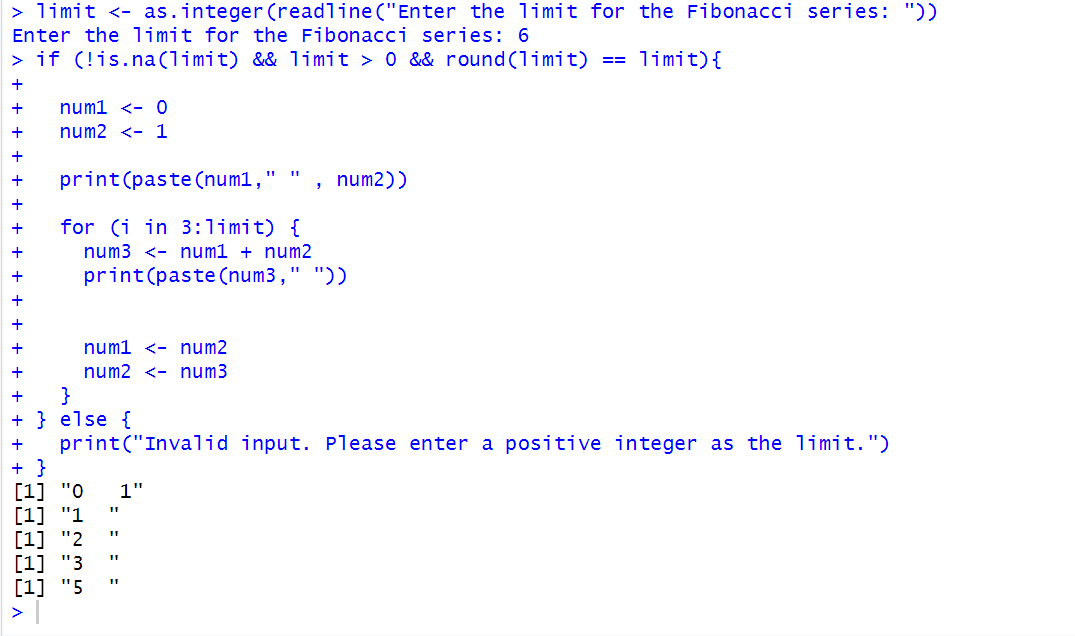
****

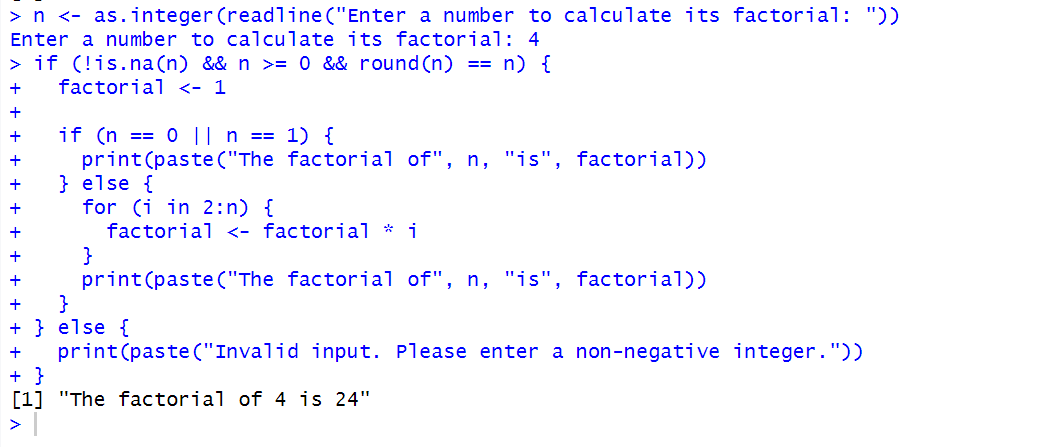
****

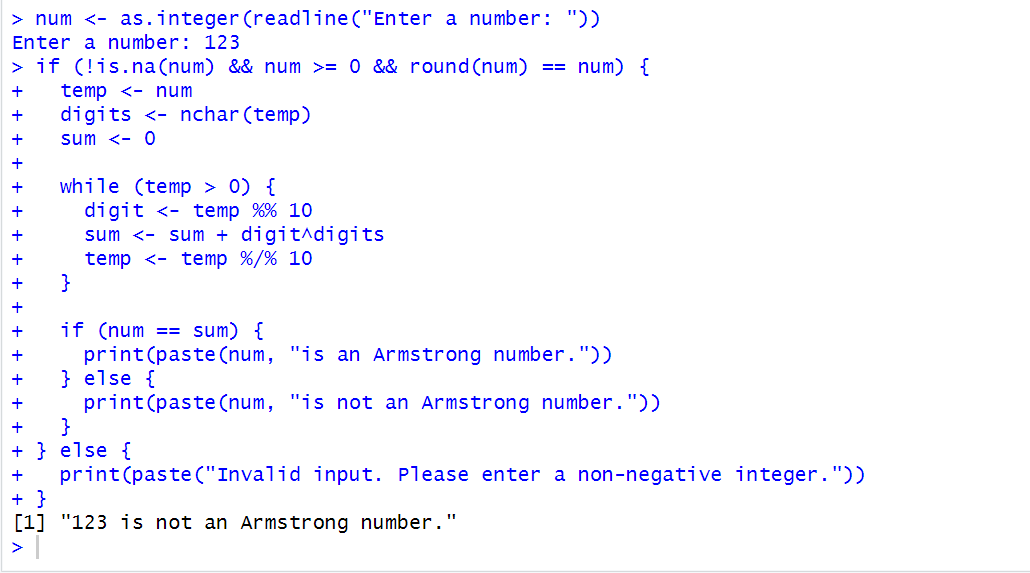
****

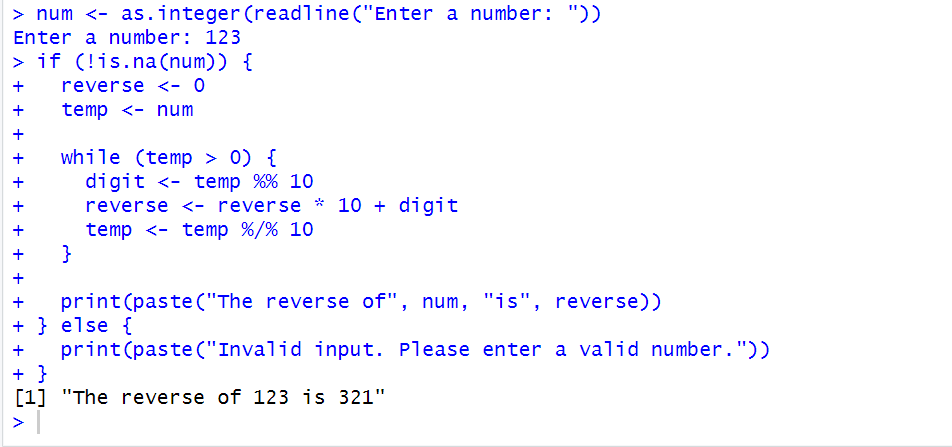
**Output:**

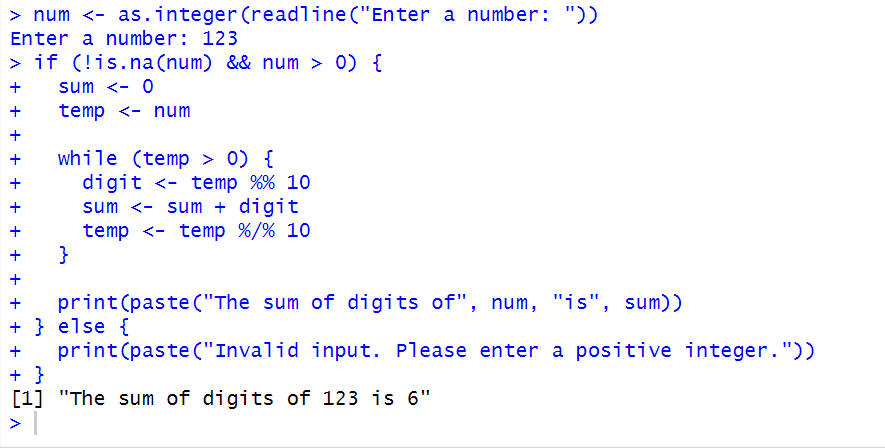


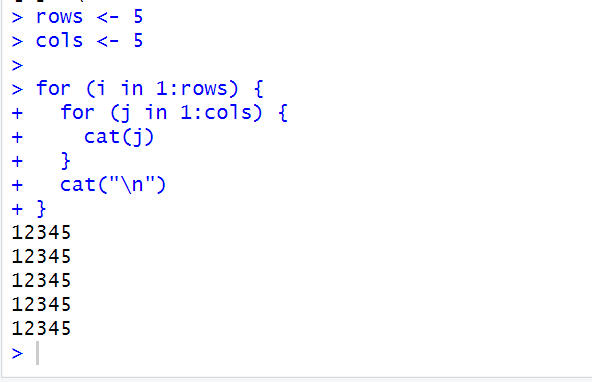
****

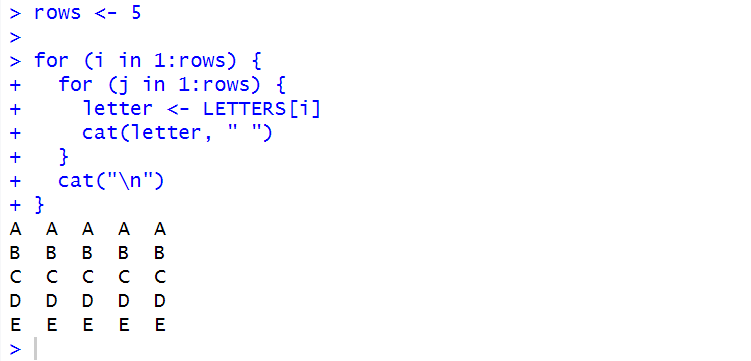
****

****

****

****

****

****

**Observation and Learnings:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.