

**ITA0480-STATISTICS WITH R PROGRAMMING FOR HEALTHCARE APPLICATIONS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No** | **TEST 2** | **MARKS** | **COs** | **BT LEVEL** |
| 1 | Given the following for() loop, predict the output and explain how the loop operates:  fruits <- c("apple", "banana", "cherry")  for (fruit in fruits) {  print(toupper(fruit))  } | 2.5 | C02 | BL4 |
| 2 | Analyze the following R code and explain what the output will be:  x <- 5  while (x > 0) {  print(x)  x <- x - 1  } | 2.5 | C01 | BL4 |
| 3 | Analyze the difference between using next and break statements in a loop. Provide an example where each would be useful. | 10 | C02 | BL3 |
| 4 | You are given a list in R that contains different types of data, including numbers, strings, and logical values:  **my\_list <- list(10, "R programming", TRUE, 3.14, "data science")**Write an R script that loops over each element in the list and performs the following actions:   * If the element is a number, multiply it by 2. * If the element is a string, convert it to uppercase. * If the element is a logical value, print "Logical value detected." | 10 | C02 | BL3 |

# 1. fruits <- c("apple", "banana", "cherry")

# for (fruit in fruits) {

# print(toupper(fruit))

# }

# [1] "APPLE"

# [1] "BANANA"

# [1] "CHERRY"

### Explanation:

1. **Initialization:**
   * The vector fruits contains three elements: "apple", "banana", and "cherry".
2. **Loop Operation:**
   * The for loop iterates over each element in the fruits vector.
   * During each iteration, the current element is assigned to the variable fruit.
3. **Iteration Details:**
   * **First Iteration:**
     + fruit is assigned the value "apple".
     + The toupper() function is called, converting "apple" to "APPLE".
     + print(toupper(fruit)) outputs "APPLE".
   * **Second Iteration:**
     + fruit is assigned the value "banana".
     + The toupper() function converts "banana" to "BANANA".
     + print(toupper(fruit)) outputs "BANANA".
   * **Third Iteration:**
     + fruit is assigned the value "cherry".
     + The toupper() function converts "cherry" to "CHERRY".
     + print(toupper(fruit)) outputs "CHERRY".

# 2.

# [1] 5

# [1] 4

# [1] 3

# [1] 2

# [1] 1

 **Initialization:**

* The variable x is initialized with the value 5.

 **Loop Condition:**

* The while loop continues to execute as long as the condition x > 0 is true.
* Since x starts at 5, the condition x > 0 is true, so the loop begins.

 **Iteration Details:**

* **First Iteration:**
  + x is 5, so the condition x > 0 is true.
  + The print(x) function outputs [1] 5.
  + The statement x <- x - 1 decreases the value of x by 1, so x becomes 4.
* **Second Iteration:**
  + x is 4, so the condition x > 0 is true.
  + The print(x) function outputs [1] 4.
  + The statement x <- x - 1 decreases the value of x by 1, so x becomes 3.
* **Third Iteration:**
  + x is 3, so the condition x > 0 is true.
  + The print(x) function outputs [1] 3.
  + The statement x <- x - 1 decreases the value of x by 1, so x becomes 2.
* **Fourth Iteration:**
  + x is 2, so the condition x > 0 is true.
  + The print(x) function outputs [1] 2.
  + The statement x <- x - 1 decreases the value of x by 1, so x becomes 1.
* **Fifth Iteration:**
  + x is 1, so the condition x > 0 is true.
  + The print(x) function outputs [1] 1.
  + The statement x <- x - 1 decreases the value of x by 1, so x becomes 0.

 **Loop Termination:**

* After the fifth iteration, x becomes 0.
* The condition x > 0 is now false, so the while loop terminates.

# 3.

### Difference Between next and break

* **next:**
  + The next statement is used to skip the current iteration of the loop and move on to the next iteration.
  + It does not terminate the loop; instead, it allows the loop to continue with the next element or condition.
* **break:**
  + The break statement is used to exit the loop entirely, regardless of the loop's condition.
  + When break is executed, the loop stops immediately, and control moves to the first statement after the loop.

# Example of next Statement

# numbers <- c(3, -1, 4, -2, 5)

# for (num in numbers) {

# if (num < 0) {

# next # Skip the current iteration if the number is negative

# }

# print(num)

# }

# [1] 3

# [1] 4

# [1] 5

# Example of break Statement

# numbers <- c(3, 7, -5, 2, 8)

# for (num in numbers) {

# if (num < 0) {

# print(paste("Negative number found:", num))

# break # Exit the loop as soon as a negative number is found

# }

# }

# [1] "Negative number found: -5"

# **Explanation:** The loop checks each number and stops as soon as it finds the first negative number -5. The break statement exits the loop immediately, so the numbers after -5 are not processed.

 Use next when you want to skip certain iterations but continue looping.

#  Use break when you want to exit the loop entirely based on a condition.

# 4.

# # Define the list

# my\_list <- list(10, "R programming", TRUE, 3.14, "data science")

# # Loop over each element in the list

# for (element in my\_list) {

# if (is. numeric(element)) {

# print(element \* 2) # Multiply numbers by 2

# } else if (is.character(element)) {

# print(toupper(element)) # Convert strings to uppercase

# } else if (is.logical(element)) {

# print("Logical value detected.") # Print message for logical values

# }

# }

# OUTPUT

# [1] 20

# [1] "R PROGRAMMING"

# [1] "Logical value detected."

# [1] 6.28

# [1] "DATA SCIENCE"

**Explanation:**

* The first element is 10, a number, so it's multiplied by 2 to give 20.
* The second element is "R programming", a string, so it's converted to uppercase, resulting in "R PROGRAMMING".
* The third element is TRUE, a logical value, so the message "Logical value detected." is printed.
* The fourth element is 3.14, a number, so it's multiplied by 2 to give 6.28.
* The fifth element is "data science", a string, so it's converted to uppercase, resulting in "DATA SCIENCE".