21/03/2023 **P.BARATH**

192121147

R programming

Assessment 2

1. Write a R program to take input from the user (name and age) and display the

values. Also print the version of R installation.

Program:

```
# Prompt the user to input their name and age

name <- readline(prompt = "Enter your name: ")

age <- readline(prompt = "Enter your age: ")


# Display the values

cat("Your name is", name, "and your age is", age, "\n")


# Print the version of R installation

cat("R version", R.version$version.string, "\n")
```

output:

```
> # Take input from the user
> name <- readline(prompt="Enter your name: ")
Enter your name: age <- readline(prompt="Enter your age: ")
>
> # Display the values entered by the user
> cat("Your name is:", name, "\n")
Your name is: age <- readline(prompt="Enter your age: ")
> cat("Your age is:", age, "\n")


>
```

2. Write a R program to get the details of the objects in memory.

Program:

```
# create some objects in memory

x <- 1:10

y <- "hello"

z <- matrix(1:9, nrow=3)


# list all objects and their sizes

objects <- ls()

for (obj in objects) {

  size <- object.size(get(obj))

  cat(obj, "size:", size, "\n")

}
```

Output:

```
> # create some objects in memory
> x <- 1:10
> y <- "hello"
> z <- matrix(1:9, nrow=3)
>
> # list all objects and their sizes
> objects <- ls()
> for (obj in objects) {
+     size <- object.size(get(obj))
+     cat(obj, "size:", size, "\n")
+ }
x size: 96
y size: 112
z size: 264
>

>
```

3. Write a R program to create a sequence of numbers from 20 to 50 and find the

mean of numbers from 20 to 60 and sum of numbers from 51 to 91.

Program:

```
# create a sequence of numbers from 20 to 50

seq_20_to_50 <- seq(20, 50)


# find the mean of numbers from 20 to 60
```

mean_20_to_60 <- mean(seq(20, 60))

 # find the sum of numbers from 51 to 91

sum_51_to_91 <- sum(seq(51, 91))

 # print the results

cat("Sequence from 20 to 50:", seq_20_to_50, "\n")

cat("Mean of numbers from 20 to 60:", mean_20_to_60, "\n")

cat("Sum of numbers from 51 to 91:", sum_51_to_91, "\n")

output:

```
> # create a sequence of numbers from 20 to 50
> seq_20_to_50 <- seq(20, 50)
>
> # find the mean of numbers from 20 to 60
> mean_20_to_60 <- mean(seq(20, 60))
>
> # find the sum of numbers from 51 to 91
> sum_51_to_91 <- sum(seq(51, 91))
>
> # print the results
> cat("Sequence from 20 to 50:", seq_20_to_50, "\n")
Sequence from 20 to 50: 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
37 38 39 40 41 42 43 44 45 46 47 48 49 50
> cat("Mean of numbers from 20 to 60:", mean_20_to_60, "\n")
Mean of numbers from 20 to 60: 40
> cat("Sum of numbers from 51 to 91:", sum_51_to_91, "\n")
Sum of numbers from 51 to 91: 2911
>
```

4. Write a R program to create a vector which contains 10 random integer v
alues
between -50 and +50.

Program:

```
# Set seed for reproducibility
set.seed(123)

# Generate vector of 10 random integers between -50 and 50
random_vector <- sample(-50:50, 10, replace = TRUE)

# Print the vector
print(random_vector)
```

output:

```
 > # Set seed for reproducibility
 > set.seed(123)
```

```
>
> # Generate vector of 10 random integers between -50 and 50
> random_vector <- sample(-50:50, 10, replace = TRUE)
>
> # Print the vector
> print(random_vector)
 [1] -20  28   0 -37  16  -9  -1  -8  50 -37
>

>
```

5. Write a R program to get all prime numbers up to a given number (based on

the sieve of Eratosthenes).

Program:

```
sieve_of_eratosthenes <- function(n) {
 # Create a boolean array "prime[1..n]" and initialize
 # all entries it as true.
 prime <- rep(TRUE, n)

 # p is initialized as 2, which is the first prime number.
 p <- 2

 while (p^2 <= n) {
  # If prime[p] is not changed, then it is a prime number.
  if (prime[p]) {
   # Update all multiples of p
   for (i in (p^2):n:p) {
    prime[i] <- FALSE
   }
  }
  p <- p + 1
 }
```

```
  # Return all prime numbers <= n
  return(which(prime)[which(prime) > 1])
}

# Example usage:
sieve_of_eratosthenes(30)
```

output:

```
> sieve_of_eratosthenes <- function(n) {
+     # Create a boolean array "prime[1..n]" and initialize
+     # all entries it as true.
+     prime <- rep(TRUE, n)
+
+     # p is initialized as 2, which is the first prime number.
+     p <- 2
+
+     while (p^2 <= n) {
+         # If prime[p] is not changed, then it is a prime number.
+         if (prime[p]) {
+             # Update all multiples of p
+             for (i in (p^2):n:p) {
+                 prime[i] <- FALSE
+             }
+         }
+         p <- p + 1
+     }
+
+     # Return all prime numbers <= n
+     return(which(prime)[which(prime) > 1])
+ }
>
> # Example usage:
> sieve_of_eratosthenes(30)
[1] 26 27 28 29 30
```

6. Write a R program to extract first 10 english letter in lower case and last 10

letters in upper case and extract letters between 22 nd to 24 th letters in upper case.

Program:

```
# create a string of letters
```

letters <- "abcdefghijklmnopqrstuvwxyz"

 # extract the first 10 letters in lower case

first_10_lower <- substr(letters, start = 1, stop = 10)

cat("First 10 letters in lower case:", first_10_lower, "\n")

 # extract the last 10 letters in upper case

last_10_upper <- toupper(substr(letters, start = nchar(letters) - 9, stop = nchar(letters)))

cat("Last 10 letters in upper case:", last_10_upper, "\n")

 # extract letters between 22nd to 24th letters in upper case

letters_between_22_to_24_upper <- toupper(substr(letters, start = 22, stop = 24))

cat("Letters between 22nd to 24th letters in upper case:", letters_between_22_to_24_upper, "\n")

output:

```
> # create a string of letters
> letters <- "abcdefghijklmnopqrstuvwxyz"
>
> # extract the first 10 letters in lower case
> first_10_lower <- substr(letters, start = 1, stop = 10)
> cat("First 10 letters in lower case:", first_10_lower, "\n")
First 10 letters in lower case: abcdefghij
>
> # extract the last 10 letters in upper case
> last_10_upper <- toupper(substr(letters, start = nchar(letters) - 9, stop
= nchar(letters)))
> cat("Last 10 letters in upper case:", last_10_upper, "\n")
Last 10 letters in upper case: QRSTUVWXYZ
>
> # extract letters between 22nd to 24th letters in upper case
> letters_between_22_to_24_upper <- toupper(substr(letters, start = 22, sto
p = 24))
> cat("Letters between 22nd to 24th letters in upper case:", letters_betwee
n_22_to_24_upper, "\n")
Letters between 22nd to 24th letters in upper case: VWX
>

>
```

7. Write a R program to find the maximum and the minimum value of a given vector.

Program:

```
# create a vector
vec <- c(23, 56, 12, 89, 45, 67, 31, 78)

# find the maximum value
max_val <- max(vec)
cat("Maximum value:", max_val, "\n")

# find the minimum value
min_val <- min(vec)
cat("Minimum value:", min_val, "\n")
```

output:

```
> # create a vector
> vec <- c(23, 56, 12, 89, 45, 67, 31, 78)
>
> # find the maximum value
> max_val <- max(vec)
> cat("Maximum value:", max_val, "\n")
Maximum value: 89
>
> # find the minimum value
> min_val <- min(vec)
> cat("Minimum value:", min_val, "\n")
Minimum value: 12
>

>
```

8. Write a R program to get the unique elements of a given string and unique

numbers of vector.

Program:

```
# create a string
str <- "hello world"

# get the unique elements of the string
unique_chars <- unique(strsplit(str, "")[[1]])
cat("Unique elements of the string:", unique_chars, "\n")

# create a vector
vec <- c(23, 56, 12, 89, 45, 67, 31, 78, 12, 23)

# get the unique numbers of the vector
unique_nums <- unique(vec)
cat("Unique numbers of the vector:", unique_nums, "\n")
```

output:

```
> # create a string
> str <- "hello world"
>
> # get the unique elements of the string
> unique_chars <- unique(strsplit(str, "")[[1]])
> cat("Unique elements of the string:", unique_chars, "\n")
Unique elements of the string: h e l o   w r d
>
> # create a vector
> vec <- c(23, 56, 12, 89, 45, 67, 31, 78, 12, 23)
>
> # get the unique numbers of the vector
> unique_nums <- unique(vec)
> cat("Unique numbers of the vector:", unique_nums, "\n")
Unique numbers of the vector: 23 56 12 89 45 67 31 78
>
```

```
>
```

9. Write a R program to create three vectors a,b,c with 3 integers. Combine the

three vectors to become a 3×3 matrix where each column represents a vector.

Print the content of the matrix.

Program:

```
a <- c(1, 2, 3)

b <- c(4, 5, 6)

c <- c(7, 8, 9)


matrix <- cbind(a, b, c)

print(matrix)
```

output:

```
> a <- c(1, 2, 3)
> b <- c(4, 5, 6)
> c <- c(7, 8, 9)
>
> matrix <- cbind(a, b, c)
> print(matrix)
     a b c
[1,] 1 4 7
[2,] 2 5 8
[3,] 3 6 9
>

>
```

10. Write a R program to create a list of random numbers in normal distribution

and count occurrences of each value.

Program:

```
 # Set the mean and standard deviation for the normal distribution

mean <- 0
```

sd <- 1

 # Generate a list of random numbers in normal distribution

random_list <- rnorm(100, mean, sd)

 # Count occurrences of each value

counts <- table(round(random_list, 2))

 # Print the counts for each value

print(counts)

output:

```
> # Set the mean and standard deviation for the normal distribution
> mean <- 0
> sd <- 1
>
> # Generate a list of random numbers in normal distribution
> random_list <- rnorm(100, mean, sd)
>
> # Count occurrences of each value
> counts <- table(round(random_list, 2))
>
> # Print the counts for each value
> print(counts)

-2.31 -1.97 -1.69 -1.55 -1.27 -1.22 -1.14 -1.12 -1.07 -1.03 -1.02 -0.95
    1     1     1     1     2     1     1     1     2     2     1     1
-0.73 -0.71 -0.69 -0.63  -0.6 -0.56  -0.5 -0.49 -0.47 -0.45  -0.4 -0.38
    1     2     3     2     1     1     1     1     2     1     1     1
-0.37 -0.35 -0.33 -0.31  -0.3 -0.28 -0.25 -0.24 -0.23 -0.22 -0.21 -0.14
    1     1     2     1     1     1     1     1     1     2     1     1
-0.08 -0.06 -0.05 -0.04 -0.03  0.01  0.05  0.11  0.12  0.15  0.18  0.22
    1     1     1     1     1     1     1     1     1     1     1     1
 0.24  0.25  0.26   0.3  0.33  0.36  0.38  0.39   0.4  0.43  0.44  0.45
    1     1     1     1     1     1     1     1     1     1     1     1
 0.46   0.5  0.55  0.58  0.64  0.69   0.7  0.78  0.82  0.84  0.88   0.9
    1     1     2     1     1     1     1     1     1     1     1     1
 0.92  0.99  1.01  1.03   1.1  1.15  1.21  1.22  1.25  1.36  1.37  1.52
    1     1     1     1     1     1     1     1     1     1     1     1
 1.53  1.79  2.05  2.17  2.19
    1     1     1     1     1
>
```

11. Write a R program to create three vectors numeric data, character data and

logical data. Display the content of the vectors and their type.

Program;

```
# Create a numeric vector

num_vector <- c(2.5, 4, 1.3, 5, 7.2)

cat("Numeric vector:\n")

print(num_vector)

cat("Type:", typeof(num_vector), "\n\n")


# Create a character vector

char_vector <- c("apple", "banana", "orange", "grape", "peach")

cat("Character vector:\n")

print(char_vector)

cat("Type:", typeof(char_vector), "\n\n")


# Create a logical vector

log_vector <- c(TRUE, FALSE, TRUE, TRUE, FALSE)

cat("Logical vector:\n")

print(log_vector)

cat("Type:", typeof(log_vector), "\n")
```

output:

```
> # Create a numeric vector
> num_vector <- c(2.5, 4, 1.3, 5, 7.2)
> cat("Numeric vector:\n")
Numeric vector:
> print(num_vector)
[1] 2.5 4.0 1.3 5.0 7.2
> cat("Type:", typeof(num_vector), "\n\n")
```

```
Type: double

>
> # Create a character vector
> char_vector <- c("apple", "banana", "orange", "grape", "peach")
> cat("Character vector:\n")
Character vector:
> print(char_vector)
[1] "apple"  "banana" "orange" "grape"  "peach"
> cat("Type:", typeof(char_vector), "\n\n")
Type: character

>
> # Create a logical vector
> log_vector <- c(TRUE, FALSE, TRUE, TRUE, FALSE)
> cat("Logical vector:\n")
Logical vector:
> print(log_vector)
[1]  TRUE FALSE  TRUE  TRUE FALSE
> cat("Type:", typeof(log_vector), "\n")
Type: logical
>
>
```

12. Write a R program to create a 5 x 4 matrix , 3 x 3 matrix with labels and fill

the matrix by rows and 2 × 2 matrix with labels and fill the matrix by columns.

Program:

```
# Create a 5 x 4 matrix filled by rows
matrix_rows <- matrix(1:20, nrow = 5, ncol = 4, byrow = TRUE)
cat("Matrix filled by rows:\n")
print(matrix_rows)

# Create a 3 x 3 matrix with row and column labels
matrix_labels <- matrix(letters[1:9], nrow = 3, ncol = 3, byrow = TRUE,
                        dimnames = list(c("Row 1", "Row 2", "Row 3"),
                                        c("Column 1", "Column 2", "Column 3")))
cat("\nMatrix with row and column labels:\n")
```

print(matrix_labels)

 # Create a 2 x 2 matrix filled by columns

matrix_columns <- matrix(c(1, 2, 3, 4), nrow = 2, ncol = 2, byrow = FALSE,

dimnames = list(c("Row 1", "Row 2"),

c("Column 1", "Column 2")))

cat("\nMatrix filled by columns:\n")

print(matrix_columns)

output:

```
> # Create a 5 x 4 matrix filled by rows
> matrix_rows <- matrix(1:20, nrow = 5, ncol = 4, byrow = TRUE)
> cat("Matrix filled by rows:\n")
Matrix filled by rows:
> print(matrix_rows)
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
[4,]   13   14   15   16
[5,]   17   18   19   20
>
> # Create a 3 x 3 matrix with row and column labels
> matrix_labels <- matrix(letters[1:9], nrow = 3, ncol = 3, byrow = TRUE,
+                         dimnames = list(c("Row 1", "Row 2", "Row 3"),
+                                         c("Column 1", "Column 2", "Colum
n 3")))
> cat("\nMatrix with row and column labels:\n")

Matrix with row and column labels:
> print(matrix_labels)
      Column 1                    Column 2 Column 3
Row 1 "abcdefghijklmnopqrstuvwxyz" NA       NA
Row 2 NA                          NA       NA
Row 3 NA                          NA       NA
>
> # Create a 2 x 2 matrix filled by columns
> matrix_columns <- matrix(c(1, 2, 3, 4), nrow = 2, ncol = 2, byrow = FALS
E,
+                          dimnames = list(c("Row 1", "Row 2"),
+                                          c("Column 1", "Column 2")))
> cat("\nMatrix filled by columns:\n")

Matrix filled by columns:
> print(matrix_columns)
      Column 1 Column 2
Row 1        1        3
Row 2        2
```

13. Write a R program to create an array, passing in a vector of values and a vector of dimensions. Also provide names for each dimension.

Program:

```
 # create a vector of values
values <- c(1, 2, 3, 4, 5, 6, 7, 8, 9)

# create a vector of dimensions
dims <- c(3, 3)

# provide names for each dimension
dimnames <- list(c("Row 1", "Row 2", "Row 3"), c("Col 1", "Col 2", "Col 3"))

# create an array
my_array <- array(values, dims, dimnames = dimnames)

# print the array
print(my_array)
```

output:

```
> # create a vector of values
> values <- c(1, 2, 3, 4, 5, 6, 7, 8, 9)
>
> # create a vector of dimensions
> dims <- c(3, 3)
>
> # provide names for each dimension
> dimnames <- list(c("Row 1", "Row 2", "Row 3"), c("Col 1", "Col 2", "Col
3"))
>
> # create an array
> my_array <- array(values, dims, dimnames = dimnames)
>
> # print the array
> print(my_array)
      Col 1 Col 2 Col 3
Row 1     1     4     7
Row 2     2     5     8
Row 3     3     6     9
>
```

14. Write a R program to create an array with three columns, three rows, and two "tables", taking two vectors as input to the array. Print the array.

Program:

```
# create two vectors
vector1 <- c(1, 2, 3, 4, 5, 6, 7, 8, 9)
vector2 <- c(10, 11, 12, 13, 14, 15, 16, 17, 18)
```

```
# create an array with three columns, three rows, and two "tables"
my_array <- array(c(vector1, vector2), dim = c(3, 3, 2))

# print the array
print(my_array)
```

output:

```
> # create two vectors
> vector1 <- c(1, 2, 3, 4, 5, 6, 7, 8, 9)
> vector2 <- c(10, 11, 12, 13, 14, 15, 16, 17, 18)
>
> # create an array with three columns, three rows, and two "tables"
> my_array <- array(c(vector1, vector2), dim = c(3, 3, 2))
>
> # print the array
> print(my_array)
, , 1

     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

, , 2

     [,1] [,2] [,3]
[1,]   10   13   16
[2,]   11   14   17
[3,]   12   15   18

>

>
```

15. Write a R program to create a list of elements using vectors, matrices and a functions. Print the content of the list.

Program:

```
# create a vector
my_vector <- c(1, 2, 3, 4, 5)

# create a matrix
my_matrix <- matrix(1:9, nrow = 3)

# create a function
my_function <- function(x) {
  return(x^2)
}

# create a list with the vector, matrix, and function
my_list <- list(my_vector, my_matrix, my_function)
```

# print the content of the list
print(my_list)

output:

```
> # create a vector
> my_vector <- c(1, 2, 3, 4, 5)
>
> # create a matrix
> my_matrix <- matrix(1:9, nrow = 3)
>
> # create a function
> my_function <- function(x) {
+     return(x^2)
+ }
>
> # create a list with the vector, matrix, and function
> my_list <- list(my_vector, my_matrix, my_function)
>
> # print the content of the list
> print(my_list)
[[1]]
[1] 1 2 3 4 5

[[2]]
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

[[3]]
function(x) {
    return(x^2)
}

>

>
```