KILLY T. MAGALLANES BSIT 2-B

1. Set up a vector named age, consisting of 34, 28, 22, 36, 27, 18, 52, 39, 42, 29, 35, 31, 27, 22, 37, 34, 19, 20, 57, 49, 50, 37, 46, 25, 17, 37, 42, 53, 41, 51, 35, 24, 33, 41.
   a. How many data points?   [1] 34

   b. Write the R code and its output.

      age <- c(34, 28, 22, 36, 27, 18, 52, 39, 42, 29,35, 31, 27, 22, 37, 34, 19, 20, 57, 49, 50, 37,
      46, 25, 17, 37, 42,53, 41,51, 35, 24, 33, 41) num_data_points
      <- length(age)
      num_data_points

      [1] 34

2. Find the reciprocal of the values for age. Write the R code and its output.
      reciprocal_age <- 1 / age reciprocal_age

      [1] 0.02941176 0.03571429 0.04545455 0.02777778
      [5] 0.03703704 0.05555556 0.01923077 0.02564103
      [9] 0.02380952 0.03448276 0.02857143 0.03225806
      [13] 0.03703704 0.04545455 0.02702703 0.02941176
      [17] 0.05263158 0.05000000 0.01754386 0.02040816
      [21] 0.02000000 0.02702703 0.02173913 0.04000000
      [25] 0.05882353 0.02702703 0.02380952 0.01886792
      [29] 0.02439024 0.01960784 0.02857143 0.04166667
      [33] 0.03030303 0.02439024

3. Assign also new_age <- c(age, 0, age).
      new_age <- c(age, 0, age)
      new_age
   What happen to the new_age?
      The new_age has 0 between the 2 ages

4. Sort the values for age.

1

Write the R code and its output. **sort(age)**
[1] 17 18 19 20 22 22 24 25 27 27 28 29 31 33 34 34 35
[18] 35 36 37 37 37 39 41 41 42 42 46 49 50 51 52 53 57

5. Find the minimum and maximum value for age.
Write the R code and its output.
   min(age)
   max(age)

   [1] 17
   [1] 57

6. Set up a vector named data, consisting of 2.4, 2.8, 2.1, 2.5, 2.4, 2.2, 2.5, 2.3, 2.5, 2.3, 2.4, and 2.7.
   data <- c(2.4, 2.8, 2.1, 2.5, 2.4, 2.2, 2.5, 2.3, 2.5, 2.3, 2.4, 2.7)
   a. How many data points? **[1] 12**

   b. Write the R code and its output.
      nmbrofdtps <- length(data)
      nmbrofdtps

      [1] 12

7. Generates a new vector for data where you double every value of the data.
   newData <- 2 * data
   newData

   What happen to the data?
   -The data doubles and was multiplied by 2

8. Generate a sequence for the following scenario:

   8.1 Integers from 1 to 100.
      Integer <- seq(1:100)
      Integer
   8.2 Numbers from 20 to 60
      Num <- seq(from=20,to=60)

Num

*8.3 Mean of numbers from 20 to 60 Num

   <- seq(from=20, to=60) xNum20to60

   <- mean(Num) xNum20to60

*8.4 Sum of numbers from 51 to 91

Number <- sum(51:91)

   Number

*8.5 Integers from 1 to 1,000

   Int <- seq(1:1000)

   Int

   a. How many data points from 8.1 to 8.4?__**143**_____

   b. Write the R code and its output from 8.1 to 8.4.

      l1 <- length(Integer)
      l2 <- length(Num)
      l3 <- length(xNum)
      l4 <- length(Number)

      total <- sum(l1, l2, l3, l4) total

   c. For 8.5 find only maximum data points until 10.
      mxdtpts <- Int[1:10]
      mxdtpts
      length(mxdtpts)

9. *Print a vector with the integers between 1 and 100 that are not divisible by 3, 5 and 7 using filter option.

   Filter(function(i) { all(i %% c(3,5,7) != 0) }, seq(100)) Write the R code and

its output.

   [1]   1 2 4 8 11 13 16 17 19 22 23 26 29 31 32 34 37
   [18] 38 41 43 44 46 47 52 53 58 59 61 62 64 67 68 71 73
   [35] 74 76 79 82 83 86 88 89 92 94 97

10. Generate a sequence backwards of the integers from 1 to 100. Write the R code and its output.

   [1] 100 99 98 97 96 95 94 93 92 91 90 89

3

```
[13] 88 87 86 85 84 83 82 81 80 79 78 77
[25] 76 75 74 73 72 71 70 69 68 67 66 65
[37] 64 63 62 61 60 59 58 57 56 55 54 53
[49] 52 51 50 49 48 47 46 45 44 43 42 41
[61] 40 39 38 37 36 35 34 33 32 31 30 29
[73] 28 27 26 25 24 23 22 21 20 19 18 17
[85] 16 15 14 13 12 11 10  9  8  7  6  5
[97]  4  3  2  1
```

11. List all the natural numbers below 25 that are multiples of 3 or 5. Find the sum of these multiples. **x <- Filter(function(i) {any(i %% c(3,5) ==0)}, seq(24))**

    Sum_of_x <- sum(x)

    Sum_of_x

    a. How many data points from 10 to 11?

       2

    b. Write the R code and its output from 10 and 11.

       datapt <- length(10:11) datapt

12. Statements can be grouped together using braces '{' and '}'. A group of statements is sometimes called a **block**. Single statements are evaluated when a new line is typed at the end of the syntactically complete statement. Blocks are not evaluated until a new line is entered after the closing brace. Enter this statement: x <- {0 + x + 5 + } Describe the output.

    #x <- {0 + x + 5 + }
    #This string has syntax error therefore it could'nt run.

13. *Set up a vector named score, consisting of 72, 86, 92, 63, 88, 89, 91, 92, 75, 75 and 77. To access individual elements of an atomic vector, one generally uses the x[i] construction. Find x[2] and x[3]. Write the R code and its output.

    score <- c(72, 86, 92, 63, 88, 89, 91, 92, 75,
    75,77.) s2 <- score[2] s2 s3 <- score[3] s3

[1] 86
 [1] 92

14. *Create a vector a = c(1,2,NA,4,NA,6,7).
    a. Change the NA to 999 using the codes print(a,na.print="-999").
    b. Write the R code and its output. Describe the output.

  a <- c(1, 2 ,NA, 4, NA, 6, 7)
  print(a,na.print="-999")

  [1]   1   2 -999   4 -999   6   7

15. A special type of function calls can appear on the left hand side of the assignment
    operator as in > class(x) <- "foo".
    Follow the codes below:
        name = readline(prompt="Input your name: ") age = readline(prompt="Input
        your age: ") print(paste("My name is",name, "and I am",age ,"years old."))
        print(R.version.string)
    What is the output of the above code?

  [1]   "My name is killy and I am 19 years old."
 > (R.version.string)
 [1] "R version 4.4.1 (2024-06-14)"