JR G. Delgado

BSIT 2A

*****Worksheet-1 in R*****

Worksheet for R Programming

Instructions:

- Use RStudio or the RStudio Cloud accomplish this worksheet. + Save the R script as *RWorksheet_lastname#1.R*.
- Create your own *GitHub repository* and push the R script as well as this pdf worksheet to your own repo.

Accomplish this worksheet by answering the questions being asked and writing the code manually.

Using functions:

seq(), assign(), min(), max(), c(), sort(), sum(), filter()

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?
   Answer: 34 data points
b. Write the R code and its output.

Code:

```
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)

age
```

Output:

```
[1] 34 28 22 36 27 18 52 39 42 29 35 31 27 22 37 34 19 20 57 49 50 37 46

[24] 25 17 37 42 53 41 51 35 24 33 41
```

2. Find reciprocal the reciprocal of the values for age.

Write the R code and its output.

Code:

```
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)
reciprocal <- 1/age
reciprocal
```

`Output:

```
[1] 0.02941176 0.03571429 0.04545455 0.02777778 0.03703704 0.05555556 0.01923077
 [8] 0.02564103 0.02380952 0.03448276 0.02857143 0.03225806 0.03703704 0.04545455
[15] 0.02702703 0.02941176 0.05263158 0.05000000 0.01754386 0.02040816 0.02000000
[22] 0.02702703 0.02173913 0.04000000 0.05882353 0.02702703 0.02380952 0.01886792
[29] 0.02439024 0.01960784 0.02857143 0.04166667 0.03030303 0.02439024
```

3. Assign also new_age <- c(age, 0, age).
   What happen to the new_age?

   Answer: display all the two age and zero its center.

Code:

```
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)
new_age <- c(age, 0, age)
new_age
```

Output:

```
[1] 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
[27] 42 53 41 51 35 24 33 41  0 34 28 22 36 27 18 52 39 42 29 35 31 27 22 37 34 19
[53] 20 57 49 50 37 46 25 17 37 42 53 41 51 35 24 33 41
```

4. Sort the values for age.
   Write the R code and its output.

Code:

```
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)

sort(age)
```

Output:

```
[1] 17 18 19 20 22 22 24 25 27 27 28 29 31 33 34 34 35 35 36 37 37 37 39 41 41 42
[27] 42 46 49 50 51 52 53 57
```

5. Find the minimum and maximum value for age.

```
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)
```

Code: max(age**)**

Output: **[1] 57**

Code: min(age)

Output: **[1] 17**

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.

a. How many data points ?

Answer: 12 data points

b. Write the R code and its Output.

Code:
```
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)
data
```

Output:
```
[1] 2.4 2.8 2.1 2.5 2.4 2.2 2.5 2.3 2.5 2.3 2.4 2.7
```

7. Generates a new vector for data where you double every value of the data. | What happen to the data?

Code:
```
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)
data*2
```

Answer: the value of your vector was been double also

8. Generate a sequence for the following scenario

8.1 Integers from 1 to 100.

```
seq(1:100)
```

8.2 Numbers from 20 to 60

```
num <- 20:60
num
```

8.3 Mean of numbers from 20 to 60

```
num <- 20:60
num
mean(num)
```

8.4 Sum of numbers from 51 to 91

```
num <- 51:91
num(num)
```

8.5 Integers from 1 to 1,000

```
num <- 1:1000
class(num)
```

a. How many data points from 8.1 to 8.4?

	8.1 = 100
	8.2 = 41
	8.3 = 41
	8.4 = 41
total:	**163 data points**

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

```
seq(1:100)
```

Output:

```
[1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
 [20]  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38
 [39]  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57
[58]  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76
[77]  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95
[96]  96  97  98  99 100
```

```
num <- 20:60
num
```

Output:

```
[1] 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
[27] 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
```

```
num <- 20:60
num
 mean(num)
```

Output:

[1] 40

```
num <- 51:91
sum(num)
```

Output:

```
[1] 2911
```

c. For 8.5 find only maximum data points until 10

code: seq(1,1000, by = 100)

Output:

```
[1]   1 101 201 301 401 501 601 701 801 901
```

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))- **error**

correct code: Filter(function(i) { all(i %% c(3,5,7) != 0) }, seq(100))-

******change small letter f to big letter F.******

Output:

```
[1]  1  2  4  8 11 13 16 17 19 22 23 26 29 31 32 34 37 38 41 43 44 46 47 52 53 58
[27] 59 61 62 64 67 68 71 73 74 76 79 82 83 86 88 89 92 94 97
```

10. Generate a sequence backwards of the integers from 1 to 100

Code:

```
num <- 100:1
num
```

Output:

```
[1] 100  99  98  97  96  95  94  93  92  91  90  89  88  87  86  85  84  83  82
[20]  81  80  79  78  77  76  75  74  73  72  71  70  69  68  67  66  65  64  63
 [39]  62  61  60  59  58  57  56  55  54  53  52  51  50  49  48  47  46  45  44
 [58]  43  42  41  40  39  38  37  36  35  34  33  32  31  30  29  28  27  26  25
```

```
[77]  24  23  22  21  20  19  18  17  16  15  14  13  12  11  10   9   8   7   6
[96]   5   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.

Code:

```
`multiple_by3 <- c(3,6,9,12,15,18,24)
multiple_by5 <- c(5,10,15,20)
sum(multiple_by3)
sum(multiple_by5)
sum(multiple_by3, multiple_by5)
```

a. How many data points from 10 to 11?

num = 100

multiple_by3 = 7

multiple_by5 = 4

total: **111 data points**

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

Code:

```
num <- 100:1
num
```

Output:

```
[1] 100  99  98  97  96  95  94  93  92  91  90  89  88  87  86  85  84  83  82
[20]  81  80  79  78  77  76  75  74  73  72  71  70  69  68  67  66  65  64  63
[39]  62  61  60  59  58  57  56  55  54  53  52  51  50  49  48  47  46  45  44
[58]  43  42  41  40  39  38  37  36  35  34  33  32  31  30  29  28  27  26  25
[77]  24  23  22  21  20  19  18  17  16  15  14  13  12  11  10   9   8   7   6
[96]   5   4   3   2   1
```

Code: multiple_by3 <- c(3,6,9,12,15,18,24)

multiple_by5 <- c(5,10,15,20)

sum(multiple_by3)

Output: **[1] 87**

sum(multiple_by5)

Output: **[1] 50**

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. **The output is error**

{ x <- 0+ x + 5 + }

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 [3]. Write the r code and its output.

Code: score <- c(72,86,92,63,88,89,91,92,75,75,77)

score[2]

Output: **[1] 86**

score[3]

Output: **[1] 92**

14. *Create a vector a = c(1,2,NA,4,NA,6,7).

Code:

a = c(1,2,NA,4,NA,6,7)

print(a,na.print="-999")

Output: [**1]   1   2 -999   4 -999   6   7**

****The NA change to -999 *****

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)
```

Code:

```
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)
```

Output:

```
[1] "My name is JR Delgado and I am 45 years old.
```