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:

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:

Input your name: lakshmi narayana

Input your age:19

[1] "My name is lakshmi narayana and I am 19 years old"

[1] "R version 4.2.2 (2022-10-31 ucrt)"

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

PROGRAM:

name = "Python";

n1 = 10;

n2 = 0.5

nums = c(10, 20, 30, 40, 50, 60)

print(ls())

print("Details of the objects in memory:")

print(ls.str())

Output:

1. “n1” “n2” “name” “nums”
2. “Details of the objects in memory:”

n1:num 10

n2:num 0.5

name: chr “Python”

nums: num[1:6] 10 20 30 40 50 60

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:

print("Sequence of numbers from 20 to 50:")

print(seq(20,50))

print("Mean of numbers from 20 to 60:")

print(mean(20:60))

print("Sum of numbers from 51 to 91:")

print(sum(51:91))

Output:

[1] "Sequence of numbers from 20 to 50:"

[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

[26] 45 46 47 48 49 50

[1] "Mean of numbers from 20 to 60:"

[1] 40

[1] "Sum of numbers from 51 to 91:"

[1] 2911

4. Write a R program to create a vector which contains 10 random integer values

between -50 and +50.

PROGRAM:

v = sample(-50:50, 10, replace=TRUE)

print("Content of the vector:")

print("10 random integer values between -50 and +50:")

print(v)

Output:

[1] "Content of the vector:"

[1] "10 random integer values between -50 and +50:"

[1] 36 32 17 45 -12 28 27 -32 -13 -29

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

the sieve of Eratosthenes).

PROGRAM:

prime\_numbers <- function(n) {

if (n >= 2) {

x = seq(2, n)

prime\_nums = c()

for (i in seq(2, n)) {

if (any(x == i)) {

prime\_nums = c(prime\_nums, i)

x = c(x[(x %% i) != 0], i)

}

}

return(prime\_nums)

}

else

{

stop("Input number should be at least 2.")

}

}

prime\_numbers(12)

Output:

1. 2 3 5 7 11

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:

print("First 10 letters in lower case:")

t = head(letters, 10)

print(t)

print("Last 10 letters in upper case:")

t = tail(LETTERS, 10)

print(t)

print("Letters between 22nd to 24th letters in upper case:")

e = tail(LETTERS[22:24])

print(e)

Output:

[1] "First 10 letters in lower case:"

[1] "Last 10 letters in upper case:"

[1] "Q" "R" "S" "T" "U" "V" "W" "X" "Y" "Z"

[1] "Letters between 22nd to 24th letters in upper case:"

[1] "V" "W" "X"

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

numbers of vector.

PROGRAM:

str1 = "The quick brown fox jumps over the lazy dog."

print("Original vector(string)")

print(str1)

print("Unique elements of the said vector:")

print(unique(tolower(str1)))

nums = c(1, 2, 2, 3, 4, 4, 5, 6)

print("Original vector(number)")

print(nums)

print("Unique elements of the said vector:")

print(unique(nums))

Output:

[1] "Original vector(string)"

[1] "The quick brown fox jumps over the lazy dog."

[1] "Unique elements of the said vector:"

[1] "the quick brown fox jumps over the lazy dog."

[1] "Original vector(number)"

[1] 1 2 2 3 4 4 5 6

[1] "Unique elements of the said vector:"

[1] 1 2 3 4 5 6

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)

m<-cbind(a,b,c)

print("Content of the said matrix:")

print(m)

Output:

[1] "Content of the said 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:

n = floor(rnorm(1000, 50, 100))

print('List of random numbers in normal distribution:')

print(n)

t = table(n)

print("Count occurrences of each value:")

print(t)

Output:

[1] "List of random numbers in normal distribution:"

[1] -79 -90 -10 93 162 -141 200 -46 -56 44 114 82 26 -151

[15] 110 83 -22 32 -99 9 88 53 131 -68 7 83 156 115

[29] 76 29 40 -57 11 223 -20 99 50 187 -48 55 231 -27

[43] 9 55 125 4 80 250 -50 236 -105 52 37 149 281 137

[57] 32 -48 99 148 122 93 144 99 131 -49 -89 0 -28 -8

[71] 134 90 -74 138 -11 -63 -15 -79 25 129 128 148 83 130

[85] -57 117 -121 140 158 19 29 17 161 -40 0 36 -109 136

[99] 79 -41 50 143 -24 -155 -1 -12 134 123 118 84 22 -51

[113] 116 27 -169 -116 89 123 -82 -63 116 -64 174 48 3 60

[127] -7 111 48 140 188 -36 -32 81 -55 -48 94 -135 149 -113

[141] 148 -36 176 147 -73 31 47 -53 311 52 156 -7 113 214

[155] 60 69 -88 -61 -30 80 -157 -51 -44 -17 23 -209 242 115

[169] 86 214 67 148 117 145 95 93 62 -7 143 -76 -90 43

[183] 103 1 31 -25 -36 209 22 -5 -63 -56 -30 63 142 179

[197] -126 48 167 195 -117 -19 54 45 85 34 140 125 6 122

[211] 62 1 -125 212 -24 110 160 -62 -2 -205 45 -35 11 -57

[225] 146 47 183 15 45 7 56 137 15 49 -44 -55 -72 113

[239] 135 94 82 86 87 34 16 82 121 -10 -101 25 -8 195

[253] 92 244 167 -35 114 156 -10 -40 18 117 -30 -14 158 -162

[267] 157 -134 53 -48 195 -38 91 39 108 179 5 105 94 -76

[281] 41 20 -17 122 -10 148 -37 -15 93 -35 -38 80 -15 73

[295] -144 134 21 91 75 65 -64 -52 142 49 43 -62 33 216

[309] 69 -43 102 110 138 221 167 -24 190 152 58 128 -15 -7

[323] 13 10 119 100 236 -130 74 -52 -19 78 163 183 -162 2

[337] -10 -64 -31 47 96 102 -3 149 148 146 -18 -33 -11 176

[351] -23 69 51 -122 -161 42 135 208 9 120 86 -50 63 143

[365] 156 191 52 177 112 103 -95 3 227 201 153 148 -207 116

[379] 26 61 18 39 3 121 -112 -33 25 13 43 31 -185 99

[393] 116 81 -53 -39 90 -15 -145 -117 68 152 104 48 137 -40

[407] 171 3 141 192 -3 79 73 150 -188 248 211 3 12 90

[421] 128 40 151 111 83 77 252 192 57 101 98 7 -74 177

[435] 56 114 -234 16 -18 32 106 264 204 148 273 -90 96 -108

[449] 101 109 85 36 -6 147 -38 205 51 -27 14 -124 73 -23

[463] -13 -39 -28 3 80 -74 216 106 52 127 -71 119 205 197

[477] -53 98 281 95 23 205 17 84 -80 8 -170 211 160 81

[491] 25 187 90 -36 -115 -47 -112 102 -4 64 244 45 103 90

[505] 26 164 35 222 165 167 -97 -7 -68 65 26 8 183 215

[519] 238 -105 -109 72 -36 -81 164 -11 107 164 0 217 121 -160

[533] 141 -35 -206 227 250 73 223 3 -52 63 106 -39 230 90

[547] 56 17 137 67 -85 115 84 160 57 140 -151 -80 -32 -100

[561] -39 170 180 -24 84 136 152 30 143 -124 20 66 135 -25

[575] 37 208 62 41 153 104 99 42 51 3 -51 -108 126 -16

[589] 51 259 22 140 -72 200 -107 66 108 -70 181 210 159 111

[603] 191 128 -131 190 -12 -52 122 79 186 68 -70 230 25 140

[617] 29 96 99 16 139 66 -117 98 72 204 60 17 -72 45

[631] 25 115 -66 -232 90 27 65 162 90 3 116 127 214 -29

[645] -1 25 79 76 135 364 -23 9 102 124 53 89 121 -19

[659] 52 -71 -9 -33 -53 63 -13 -53 101 -155 308 -43 1 133

[673] 83 -10 190 145 10 154 36 26 -2 296 143 49 -4 138

[687] -46 -189 -68 75 87 137 -112 180 136 11 -14 160 93 84

[701] 70 54 19 154 -2 -44 88 108 76 8 -74 -14 132 4

[715] -105 73 163 195 26 293 29 -129 -54 123 164 140 34 -76

[729] 189 89 -9 11 -16 69 -137 202 203 -35 52 7 212 43

[743] -28 -116 156 173 127 225 44 48 79 92 -15 180 76 173

[757] 221 211 147 -100 55 133 161 -21 -20 177 -70 102 -59 191

[771] 130 -81 118 69 18 30 12 89 13 113 -126 -51 22 114

[785] 58 47 -188 54 -51 157 41 -118 119 132 88 73 -102 137

[799] 94 47 163 217 6 156 55 0 -18 18 12 39 6 119

[813] -49 5 127 130 49 72 68 160 108 78 42 -55 110 40

[827] 12 15 74 -21 73 129 -48 168 262 141 67 18 229 61

[841] 16 74 133 -6 31 134 6 56 139 61 -38 24 51 -20

[855] 39 -37 77 114 24 14 28 -92 6 177 17 52 122 140

[869] -68 85 121 -5 88 190 98 16 169 -118 70 83 56 30

[883] -146 201 65 -172 47 214 -37 83 54 -131 21 -67 115 -59

[897] -71 86 -112 10 209 -84 86 40 87 78 6 -25 60 36

[911] 275 165 185 -6 48 -60 -6 54 36 55 266 -2 0 -31

[925] 176 30 55 182 -54 -84 47 -9 311 264 -26 138 109 43

[939] 83 185 154 58 57 -89 5 -6 157 -60 58 -41 -195 187

[953] -19 -50 -59 178 -60 125 42 -19 121 136 -54 -36 -137 115

[967] 26 122 158 -34 24 64 17 216 168 96 -121 149 -18 142

[981] 50 12 -8 27 30 138 -28 68 48 82 140 72 -201 10

[995] 61 -59 97 66 33 3

[1] "Count occurrences of each value:"

n

-234 -232 -209 -207 -206 -205 -201 -195 -189 -188 -185 -172 -170 -169 -162

1 1 1 1 1 1 1 1 1 2 1 1 1 1 2

-161 -160 -157 -155 -151 -146 -145 -144 -141 -137 -135 -134 -131 -130 -129

1 1 1 2 2 1 1 1 1 2 1 1 2 1 1

-126 -125 -124 -122 -121 -118 -117 -116 -115 -113 -112 -109 -108 -107 -105

2 1 2 1 2 2 3 2 1 1 4 2 2 1 3

-102 -101 -100 -99 -97 -95 -92 -90 -89 -88 -85 -84 -82 -81 -80

1 1 2 1 1 1 1 3 2 1 1 2 1 2 2

-79 -76 -74 -73 -72 -71 -70 -68 -67 -66 -64 -63 -62 -61 -60

2 3 4 1 3 3 3 4 1 1 3 3 2 1 3

-59 -57 -56 -55 -54 -53 -52 -51 -50 -49 -48 -47 -46 -44 -43

4 3 2 3 3 5 4 5 3 2 5 1 2 3 2

-41 -40 -39 -38 -37 -36 -35 -34 -33 -32 -31 -30 -29 -28 -27

2 3 4 4 3 6 5 1 3 2 2 3 1 4 2

-26 -25 -24 -23 -22 -21 -20 -19 -18 -17 -16 -15 -14 -13 -12

1 3 4 3 1 2 3 5 4 2 2 6 3 2 2

-11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3

3 6 3 3 5 5 2 2 2 4 2 5 3 1 10

4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

2 3 6 4 3 4 4 4 5 3 2 3 5 6 5

19 20 21 22 23 24 25 26 27 28 29 30 31 32 33

2 2 2 4 2 3 7 7 3 1 4 5 4 3 2

34 35 36 37 39 40 41 42 43 44 45 47 48 49 50

3 1 5 2 4 4 3 4 5 2 5 7 7 4 3

51 52 53 54 55 56 57 58 60 61 62 63 64 65 66

5 7 3 5 6 5 3 4 4 4 3 4 2 4 4

67 68 69 70 72 73 74 75 76 77 78 79 80 81 82

3 4 5 2 4 7 3 2 4 2 3 5 4 3 4

83 84 85 86 87 88 89 90 91 92 93 94 95 96 97

8 5 3 5 3 4 4 8 2 2 5 4 2 4 1

98 99 100 101 102 103 104 105 106 107 108 109 110 111 112

4 6 1 3 5 3 2 1 3 1 4 2 4 3 1

113 114 115 116 117 118 119 120 121 122 123 124 125 126 127

3 5 6 5 3 2 4 1 6 6 3 1 3 1 4

128 129 130 131 132 133 134 135 136 137 138 139 140 141 142

4 2 3 2 2 3 4 4 4 6 5 2 9 3 3

143 144 145 146 147 148 149 150 151 152 153 154 156 157 158

5 1 2 2 3 8 4 1 1 3 2 3 6 3 3

159 160 161 162 163 164 165 167 168 169 170 171 173 174 176

1 5 2 2 3 4 2 4 2 1 1 1 2 1 3

177 178 179 180 181 182 183 185 186 187 188 189 190 191 192

4 1 2 3 1 1 3 2 1 3 1 1 4 3 2

195 197 200 201 202 203 204 205 208 209 210 211 212 214 215

4 1 2 2 1 1 2 3 2 2 1 3 2 4 1

216 217 221 222 223 225 227 229 230 231 236 238 242 244 248

3 2 2 1 2 1 2 1 2 1 2 1 1 2 1

250 252 259 262 264 266 273 275 281 293 296 308 311 364

2 1 1 1 2 1 1 1 2 1 1 1 2 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:

a = c(1, 2, 5, 3, 4, 0, -1, -3)

b = c("Red", "Green", "White")

c = c(TRUE, TRUE, TRUE, FALSE, TRUE, FALSE)

print(a)

print(typeof(a))

print(b)

print(typeof(b))

Print(c)

print(typeof(c))

Output:

[1] 1 2 5 3 4 0 -1 -3

[1] "double"

[1] "character"

[1] TRUE TRUE TRUE FALSE TRUE FALSE

[1] "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:

m1 = matrix(1:20, nrow=5, ncol=4)

print("5 × 4 matrix:")

print(m1)

cells = c(1,3,5,7,8,9,11,12,14)

rnames = c("Row1", "Row2", "Row3")

cnames = c("Col1", "Col2", "Col3")

m2 = matrix(cells, nrow=3, ncol=3, byrow=TRUE, dimnames=list(rnames, cnames)) print("3 × 3 matrix with labels, filled by rows: ")

print(m2)

print("3 × 3 matrix with labels, filled by columns: ")

m3 = matrix(cells, nrow=3, ncol=3, byrow=FALSE, dimnames=list(rnames, cnames)) print(m3)

Output:

[1] "5 × 4 matrix:"

[,1] [,2] [,3] [,4]

[1,] 1 6 11 16

[2,] 2 7 12 17

[3,] 3 8 13 18

[4,] 4 9 14 19

[5,] 5 10 15 20

[1] "3 × 3 matrix with labels, filled by rows: "

Col1 Col2 Col3

Row1 1 3 5

Row2 7 8 9

Row3 11 12 14

[1] "3 × 3 matrix with labels, filled by columns: "

Col1 Col2 Col3

Row1 1 7 11

Row2 3 8 12

Row3 5 9 14

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:

a = array( 6:30, dim = c(4, 3, 2), dimnames = list( c("Col1", "Col2", "Col3", "Col4"), c("Row1", "Row2", "Row3"), c("Part1", "Part2")

)

)

print(a)

Output:

, , Part1

Row1 Row2 Row3

Col1 6 10 14

Col2 7 11 15

Col3 8 12 16

Col4 9 13 17

, , Part2

Row1 Row2 Row3

Col1 18 22 26

Col2 19 23 27

Col3 20 24 28

Col4 21 25 29

14. Write a R program to create an array with three columns, three rows, and two

&quot;tables&quot;, taking two  vectors as input to the array.  Print the array

PROGRAM:

v1 = c(1, 3, 5, 7)

v2 = c(2, 4, 6, 8, 10)

arra1 = array(c(v1, v2),dim = c(3,3,2))

print(arra1)

Output:

, , 1

[,1] [,2] [,3]

[1,] 1 7 6

[2,] 3 2 8

[3,] 5 4 10

, , 2

[,1] [,2] [,3]

[1,] 1 7 6

[2,] 3 2 8

[3,] 5 4 10

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

functions. Print the content of the list.

PROGRAM:

l = list( c(1, 2, 2, 5, 7, 12), month.abb, matrix(c(3, -8, 1, -3), nrow = 2), asin )

print("Content of the list:")

print(l)

Output:

[1] "Content of the list:"

[[1]]

[1] 1 2 2 5 7 12

[[2]]

[1] "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep" "Oct" "Nov" "Dec"

[[3]]

[,1] [,2]

[1,] 3 1

[2,] -8 -3

[[4]]

function (x) .Primitive("asin")