**Module 1 Homework**

**Problem 1 (32 points)** Choose the answers in the following questions:

**(a)** What is the class of the object defined be vec <-c(5,TRUE) ?

• Numeric

• Integer

• Matrix  
• Logical

Answer) **Numeric.**

> vec<-c(5,TRUE)

> class(vec)

[1] "numeric"

**(b)** Suppose I have vectors x <- 1:4 and y <- 1:2. What is the result of the expression x + y?

• A numeric vector with the values 1, 2, 5, 7

• A numeric vector with the values 2, 4, 2, 4

• An integer vector with the values 2, 4, 4, 6

• An error

Answer) **An integer vector with the values 2, 4, 4, 6**

**> x<-1:4**

**> y<-1:2**

**> x+y**

**[1] 2 4 4 6**

**(c)** What is returned by the R command c(1,2) %\*% t(c(1,2)) ?

• The number 5  
• A one by two matrix  
• A two by two matrix

• An error is returned because the dimensions mismatch

Answer) **A two by two matrix**

> c(1,2)%\*%t(c(1,2))

[,1] [,2]

[1,] 1 2

[2,] 2 4

**(d)** Suppose I define the following function in R:

f <- function(x) {

g <- function(y) {

y+z

}

z<-4

x+g(x)

}  
If I then run in R the following statements

z<-15

f(3)

What value is returned?

• 16

•7

• 10

•4

Answer) **10**

**> f <- function(x) {**

**+ g <- function(y) {**

**+ y+z**

**+ }**

**+ z<-4**

**+ x+g(x)**

**+ }**

**> f(3)**

**[1] 1**

**Problem 2 (10 points)**

Use R to calculate =

Please hand in your R commands and the results you produce by running those commands.

**Answer:**

**> x<-1:1000**

**> sum(x^2)**

**Output:**

**[1] 333833500**

**Question 3 (18 points)**

This exercise is to make sure all of you understand how to create a vector in R and do simple operations. **All parts should be done using “R”** obviously.

Consider a group of 10 randomly selected people of **different** ages.

1. Create a vector named “age” to represent this. You can pick any **reasonable** age (whole numbers only please) for each person.

Answer)

**> age<-c(12, 14, 16, 18, 20, 22, 24, 25, 27, 29)**

**> age**

**Output:**

[1] 12 14 16 18 20 22 24 25 27 29

1. Multiply each person’s age by 12 (to convert into months). (the answer should be a vector, hope you know this)

Answer)

> age\*12

Output:

[1] 144 168 192 216 240 264 288 300 324 348

1. Find the sum of ages of all these people.

**Answer)**

**> sum(age)**

**Output:**

**[1] 207**

1. Find the age of the youngest person

**Answer)**

**> min(age)**

**Output:**

**[1] 12**

1. Find the age of the oldest person.

**Answer)**

**> max(age)**

**Output:**

**[1] 29**

1. Find the square root of the age of each person. (Not sure what this means, but who cares?) (this also should be a vector)

**Answer)**

**> sqrt(age)**

**Output:**

**[1] 3.464102 3.741657 4.000000 4.242641**

**[5] 4.472136 4.690416 4.898979 5.000000**

**[9] 5.196152 5.385165**

**Question 4 (40 points)**

Write an R script that does all of the following:

1. Create a vector X of length 30, with the kth element in X = 3k, for k=1…30. Print out the values of X.

**Answer)**

**> X1<-seq(1,30)**

**> X<-3\*X1**

**> X**

**Output:**

**[1] 3 6 9 12 15 18 21 24 27 30 33 36 39**

**[14] 42 45 48 51 54 57 60 63 66 69 72 75 78**

**[27] 81 84 87 90**

1. Create a vector Y of length 30, with all elements in Y equal to 0. Print out the values of Y.

**Answer)**

**> Y<-rep(0,30)**

**> Y**

**Output:**

**[1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0**

**[21] 0 0 0 0 0 0 0 0 0 0**

1. Using a “for” loop, reassigns the value of the k-th element in Y, for k = 1…30. When k < 20, the kth element of Y is reassigned as the sine of (2k). When the k ≥ 20, the kth element of Y is reassigned as the value of integral . (You may want to use $value at the end of the line to get the integration with R clean out unwanted values)

Please run the script and hand in your R execution results. The R script file should be submitted separately as part of the “hw1.r” file.

**Answer)**

**for (k in 1:30) {**

**if (k < 20) {**

**Y[k] <- sin(2\*k)**

**} else {**

**Y[k] <- integrate(function(t) sqrt(t), lower=0, upper=k)$value**

**}**

**}**

**print(Y)**

**Output:**

[1] 0.909297427 -0.756802495 -0.279415498 0.989358247 -0.544021111 -0.536572918

[7] 0.990607356 -0.287903317 -0.750987247 0.912945251 -0.008851309 -0.905578362

[13] 0.762558450 0.270905788 -0.988031624 0.551426681 0.529082686 -0.991778853

[19] 0.296368579 59.628486093 64.156066931 68.792772200 73.536091612 78.383680568

[25] 83.333342688 88.383014823 93.530754108 98.774726701 104.113197958 109.544523798