Part A: Data types

1. What will be the results of these statements?

(a) class("GINGER")

```
[1] "character"
```

(b) class(TRUE)

```
[1] "logical"
```

(c) class(NaN)

```
[1] "numeric"
```

2. How can you coerce the following expressions, and what value will it return?

(a) 1 to type logical

```
as.logical(1)

[1] TRUE
```

(b) "12.3" to type numeric

```
as.numeric("12.3")

[1] 12.3
```

(c) "GINGER" to type numeric

```
as.numeric("GINGER")

Warning: NAs introduced by coercion

[1] NA
```

Part B: Data structures

Vectors

1. How can you create a vector with the following elements: 1,2,3,4,5 ?

```
c(1,2,3,4,5)

[1] 1 2 3 4 5
```

2. Create the same vector using a different approach.

```
1:5

[1] 1 2 3 4 5
```

3. Given the following vector: x <- c("first", "second", "third"), what will the following expressions return?

```
length(x)
[1] 3
x[3]
[1] "third"
is.na(x)
[1] FALSE FALSE FALSE
```

4. What will the following vector creation statements return? And what will their type be?

(a) c(1,2,"3",NA)

```
c(1,2,"3",NA)
[1] "1" "2" "3" NA
typeof(c(1,2,"3",NA))
[1] "character"
```

(b) c(1L,NA,3L)

```
c(1L,NA,3L)
[1]  1 NA  3
typeof(c(1L,NA,3L))
[1] "integer"
```

(c) as.logical(c(TRUE, FALSE, 0, 23))

```
as.logical(c(TRUE, FALSE, 0, 23))
[1]  TRUE FALSE FALSE  TRUE
typeof(as.logical(c(TRUE, FALSE, 0, 23)))
[1] "logical"
```

(d) c(NaN, 12.3, Inf)

```
c(NaN, 12.3, Inf)
[1]  NaN 12.3  Inf
typeof(c(NaN, 12.3, Inf))
[1] "double"
```

Matrices

1. What code do you need to write to create the following matrix m?

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

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

```
m <- matrix(1:9, nrow=3, ncol=3,byrow = TRUE)
```

2. Given the 3x3 matrix myou created in (1), how can you get its third row?

```
m[3,]
```

```
[1] 7 8 9
```

3. What will the code below return?

```
m[1,2]
```

```
[1] 2
```

Lists

1. Create a list l with the following elements: 1, “GINGER”, TRUE

```
l <- list(1, "GINGER", TRUE)
```

2. How can you access its second element (“GINGER”)?

```
l[[2]]
```

```
[1] "GINGER"
```

3. You want to be able to access its elements using l$first, l$second and l$third. What do you need to do?

```
names(l) = c("first", "second", "third")
```

4. How could you create the same list l with named elements directly?

```
l <- list(first=1, second="GINGER", third=TRUE)
```

Data Frames

1. Create the following data frame df:

ids	sex	age
x1	female	20
x2	male	34
x3	female	23

```
df <- data.frame(
 id=c("x1", "x2", "x3"),
 sex=c("female", "male", "female"),
 age=c(20, 34, 23)
)
```

2. What is the type of the column id in the data frame df you created in (1)?

```
[1] "integer"
```

3. How can you create the same data frame but making sure that id is a character?

```
df <- data.frame(
 id=c("x1", "x2", "x3"),
 sex=c("female", "male", "female"),
 age=c(20, 34, 23),
 stringsAsFactors = FALSE
)
```

4. What are the dimensions of the data frame you just created?

3 rows x 3 columns

5. What will the following statement return?

```
df["age", 3]
```

```
[1] NA
```

6. Why?

The expression above is trying to access the element at row “age” and column 3, which doesn’t exist (we haven’t even named the rows). It is likely that the desired correct statement was df[3, "age"] or df$age[3], which would have returned the value of the column age of the third row (23).