# Part 4: Questions to Answer

Replace 'write your answer here' with your responses. Be sure your answers have been 'highlighted' using the triple hash `###` which makes the text large and bold.

1. Coercion: For each of the following, explain what type of output you will receive and why R is producing that output.

a. `c(0, TRUE)`

b. `c("F", F)`

c. `c(list(1), "b")`

d. `c(FALSE, 1L)`

1. **It would produce 🡪 [1] 0 1 because R notes TRUE = 1 & FALSE = 0. Since it was in vector format, it coerced it into the least restrictive type.**
2. **It would product 🡪 [1] "F" "FALSE" because “F” is left in quotation marks and signifies it as a character. F is the default for FALSE, but again R coerces the vector into the least restrictive type.**

**c) It would produce 🡪 [[1]] [1] 1**

**[[2]]**

**[1] "b" because you’ve called the vector to become a list under a vector. Therefore it created a list under the vector.**

**d) It would produce 🡪 [1] 0 1 because R notes TRUE = 1 & FALSE = 0. Since it was in vector format, it coerced it into the least restrictive type.**

2. What is the difference between NULL, NA, and NaN?

**NA in an atomic vector of matrix will not change the data type. Adding in NA will not change the types of data because it counts as missing and there is a type of NA for each data. NA is used to represent missing or unknown values.**

**NULL is used to represent an empty or nonexistent value. NA is a default element already in R, and it is the most logical type. NULL is ready to give a TRUE/FALSE answer, but no answer has been given. NULL is like “nothing” there.**

**NaN is used to represent indeterminate forms in mathematics.**

3. What is the difference between logical(0) and NULL? Write a command (other than `logical(0)`) that will produce logical(0) as the output. Write a command (other than `NULL`) that will produce NULL as the output.

**NULL means that a specific object is missing, and nothing is there. Logical(0) means that there is no TRUE/FALSE answer given and is missing.**

4. A vector `c(TRUE, FALSE)` is a logical vector. Other than `TRUE` or `FALSE`, what can you insert into the vector so that it increases to a length of 3 and remains a logical vector and does not get coerced into another class?

**You could add NA to increase the length but does not coerce it into another class besides logical.**

5. What are the lengths of the following lists? Use bracket notation to subset them to the letters "h" and "i". Be sure to print the result so it shows the subset.

l1 <- list(letters[1:5], letters[3:9] , letters[4:7])

l1

l2 <- list( c(letters[1:5], letters[3:9]), letters[4:7] )

l2

```

**Length(l1) = 3, length(l2) = 2**

**print(c(l1[[2]][6], l1[[2]][7]))**

**print(c(l2[[1]][11], l2[[1]][12]))**

6. What will `c(4:7) \* c(2:4)` produce? Briefly, why?

**[1] 8 15 24 14 because in the Warning Message it states, “**In c(4:7) \* c(2:4) :longer object length is not a multiple of shorter object length” This means that c(4:7) has a length of 4 and c(2:4) has a length of 3. It would multiply it by element-wise until it reaches the last element. Then it would start from the beginning and multiply the latest element with the next beginning element. Hence, 7 \* 2 = 14.

7. Take a look at the following code chunks. What are some of the differences between `cat()` and `print()`?

```{r, error = TRUE}

cat(letters[1:3], letters[24:26])

print(letters[1:3], letters[24:26]) # Why are we getting the following error?

# Error in print.default(letters[1:3], letters[24:26]) : invalid 'digits' argument

cat(l1)

print(l1)

```

**Print() function gives a value as a vector, list, etc. inside of it. It can only print one value per function. Meanwhile, cat() doesn’t give the value as a vector, list, etc. Cat() is also meant for atomic types and names. This means you cannot call any empty vectors. The reason why there is an error for print() is because it can only print one value at a time, as compared to cat() function (distinguished by the comma).**

8. What happens to a factor when you reverse its levels?

```{r}

f1 <- factor(c("A","A","B","C","D","A","C"))

f1

levels(f1) <- rev(levels(f1))

f1

```

**If you reverse just the levels for this factor problem, you will have the result match the levels for the reverse. For instance, without reversal, the first 3 results would be “A” “A” “B” for levels A, B, C, D. Now if you reverse the levels for D, C, B, A, then the results will be replaced with the new level reversal as “D” “D” “C”. The result stays the same with respect to the levels.**

9. How do f2 and f3 differ from the unmodified f1?

```{r}

f1 <- factor(c("A","A","B","C","D","A","C"))

f1

f2 <- factor(rev(c("A","A","B","C","D","A","C")))

f2

f3 <- factor(c("A","A","B","C","D","A","C"), levels = rev(c("A","B","C","D")))

f3

```

**For f2, it is only reversing the order of the vector, while the levels stay ordered alphabetically. For f3, the vector remains the same result as f1, but the levels of f3 is being reversed.**

10. What attributes does a data frame possess?

**It has names (including col.names & row.names) and class.**

11. What does as.matrix() do when applied to a data frame with columns of different types? Create a simple data.frame with two columns: one numeric and one string. Use as.matrix and show the results.

**df1 <- data.frame("x" = letters[1:5], "y" = c(1, 2, 3, 4, 5))**

**df1**

**df2 <- as.matrix(df1)**

**typeof(df2)**

**It coerces all other types to the least restrictive type. In this case, it coerced everything into a character matrix.**