EDS 221 Review Week 1

Code execution

Problem 1

Briefly describe what each of the following operators/functions does.

```
:
<-
mean()
```

Problem 2

Evaluate the following code. After it runs:

What are the values of x and y?

What output would you see on the console?

```
x <- seq(3, 12, by = 3)
y <- x - 3
sqrt(x + y)
length(x + y)
```

Problem 3

Consider the following code.

```
coral_genera <- c("Porites", "Acropora", "Pocillopora")
coral_sample <- sample(coral_genera, size = 2, replace = FALSE)
```

Which of the following is a possible value of `coral_sample`?

```
"Porites" "Acropora"
```

```
"Porites"
```

```
"Porites" "Porites"
```

Data structures

Problem 1

Each of the following expressions evaluates to one of the four atomic types. Identify which is which.

```
2:10
2 / 10
c("2.0", "10.0")
2 > 10
```

Problem 2

Evaluate the following code. After it runs, what is the value of `is_time_travel`? What is the type of `combined`?

```
now <- 2026
then <- 2000
is_time_travel <- then > now
combined <- c(now, then, is_time_travel)
```

Problem 3

Consider the following matrix:

```
bird_counts_by_site <- matrix(c(6L, 3L, 7L, 10L, 9L, 0L), nrow = 3)
bird_counts_by_site
```

```
     [,1] [,2]
[1,]    6   10
[2,]    3    9
[3,]    7    0
```

Which of the following expressions *won't* return the numbers 6, 3, and 7?

```r
bird_counts_by_site[, 1]
```

```r
bird_counts_by_site[1, ]
```

```r
bird_counts_by_site[1:3, 1]
```

Control structures

Problem 1

Consider each of the following claims. Indicate whether each one is true or false.

`if` statements must be followed by `else`

`else` statements must follow `if`

Only one `else if` statement is allowed in a conditional statement

Problem 2

Read the following code. What value in the blank would make the `else` block run?

```r
species_endangered <- c(leatherback = TRUE, hawksbill = TRUE, green = FALSE)
species <- "______"

if (species_endangered[species]) {
  print(paste("Species", species, "is endangered"))
} else {
  print(paste("Species", species, "is not endangered"))
}
```

Problem 3

Assume this code runs first.

```
species_endangered <- c(leatherback = TRUE, hawksbill = TRUE, green = FALSE)
turtle_pop <- c(leatherback = 30000, hawksbill = 20000, green = 85000)
endangered_pop <- 0
```

How would you edit this loop to add up the endangered turtle populations?

```
for (t in turtle_pop) {
  if (species_endangered[t]) {
    endangered_pop <- endangered_pop + turtle_pop[t]
  }
}
```

Functions

Problem 1

In the following code, indicate where to find:

- The function keyword
- A parameter
- An argument
- The return value

```
f_to_c <- function(deg_f) {
  deg_c <- (deg_f - 32) * 5 / 9
  return(deg_c)
}

f_to_c(32)
```

Problem 2

Fill in the blanks to complete this function.

```
initialize_reef <- function(____, content) {
  new_reef <- matrix(content, nrow = size_reef, ncol = size_reef)
  ____(new_reef)
}
```

Problem 3

Running this code:

```
roll_dice <- function(n_dice, n_sides) {
  roll <- sum(sample(1:n_sides, size = n_dice, replace = TRUE))
  return(roll)
}

coral_fate <- function(roll, mort_thr, grow_thr) {
  if (roll <= mort_thr) {
    fate <- "death :("
  } else if (roll >= grow_thr) {
    fate <- "growth :)"
  } else {
    fate <- "survival :|"
  }
  return(fate)
}

roll2d6 <- roll_dice(2, n_sides = 6)
coral_fate(roll2d6)
```

Yields this error:

```
Error in `coral_fate()`:
! argument "mort_thr" is missing, with no default
```

How would you correct this code?