Stats 102A Old Practice Midterm

This is a practice exam based on questions from a past midterm exam. The actual exam will have different questions.

The exam will cover content from:

- Lecture 1-2 Basic Data Structures
- Lecture 1-3 Subsetting
- Lecture 2-1 Conditions, Loops
- Lecture 2-2 Functions
- Lecture 2-3 Environments and scoping
- Lecture 3-2 dplyr
- Lecture 4-1 dplyr / tidy
- Lecture 4-2 Regular Expressions, the following cheat-sheet will be attached to the exam: https://cheatography.com/davechild/cheat-sheets/regular-expressions/pdf/
- Lecture 5-1 S3, no S4 or RC on midterm exam
- Lecture 5-2 R6, no private or active fields on midterm exam
- Lecture 3-1 importing/exporting/lubridate/rvest will not be covered on the midterm exam

You must not share or distribute this practice exam without permission.

You may upload this practice exam and your lecture notes to ChatGPT (or other AI tool) and ask it to create additional practice questions.

This practice exam is free response. The midterm exam will be multiple choice.

Exam Instructions:

Each question shows some code. Your job is to read the code and write down what R will output.

If the code will result in an error, you must indicate that it will result in an error, but you do not need to specify the exact error message.

If the code will result in a warning, you must indicate that it will result in a warning, but you do not need to specify the exact warning message.

If `[1]` or `[[1]]` or `[1,]` is part of the output, make sure you include it in the output.

For each numbered problem, assume you are starting with an empty global environment. For parts within a single numbered problem, the objects created continue to exist. The environment resets with each **new** numbered problem.

1)

```
f <- function(y) {
  y <- y + 1
  2 * x
  }
x <- 3
y <- 5
f(x)
```

2)

```
a <- 4:1
a + 3:5
```

3)

```
junk <- list(c(TRUE, "FALSE"), c(FALSE, TRUE))
typeof(junk[1])
```

```
junk[[1]][1]
```

```
junk[2][2]
```

```
typeof(junk[2][2])
```

4)

```
x <- list(-5, 3L, FALSE, NA, NULL)
for (i in x) {
  print(typeof(i))
}
```

5)

```
d <- c(1, 2, NULL, 4)
g <- function(a) {
  if (a %%2 == 0) {
    even <- TRUE
  } else {
    even <- FALSE
  }
  return(even)
}
g(d)
```

```
for (i in d) { print(g(i)) }
```

6)

```
x <- matrix(1:12, nrow = 4)
x[3, ]
```

```
x[3, 2, drop = FALSE]
```

7)

```
df <- data.frame(
    x = 1:3,
    y = c("a", "b", "c"),
    stringsAsFactors = FALSE
  )
typeof(df)
```

```
class(df)
```

```
df[1]
```

```
df[[1]]
```

```
df[1,]
```

8)

```
library(stringr)
greedy_pattern <- "[Aa]\\w*a"
ungreedy_pattern <- "[Aa]\\w*?a"
string <- "Alabama, Alaska, California"
str_extract_all(string, greedy_pattern)
```

```
str_extract_all(string, ungreedy_pattern)
```

9)

```
x <- 1; y <- 2; z <- 3
h <- function() {
    y <- 4
    j <- function() {
         print(c(x, y, z))  # note this print statement
         return(x + y + z)
         }
    return(j())
  }
h()
```

10)

```
x <- 1; y <- 2; z <- 3
h <- function(){
    j <- function(){
         y <- 4
         return(x + y + z)
         }
    print(c(x, y, z))  # note this print statement
    return(j())
  }
h()
```

11)

```
x <- 1; y <- 2; z <- 3
h <- function(){
    j <- function(){
         y <<- 4        # super assignment
         return(x + y + z)
         }
    result <- j()    # j() is run
    print(c(x, y, z))  # note this print statement
    return(result)
  }

h()
```

12) Look at the following data.frame, `cases1`. Use tidyr to reshape the data. The first example where the column selection is 2:4 has been done for you. What will R output if the selection column is 3:4?

```
library(tidyr)
cases1
  country 2011 2012 2013
1      FR 7000 6900 7000
2      DE 5800 6000 6200

pivot_longer(cases1, "year", "n", 2:4)
  country year    n
1      FR 2011 7000
2      DE 2011 5800
3      FR 2012 6900
4      DE 2012 6000
5      FR 2013 7000
6      DE 2013 6200

pivot_longer(cases1, "year", "n", 3:4)
```

13)

```
f <- function(x) { UseMethod("f") }
f.j <- function(x) { x + 3 }
f.k <- function(x) { x + 4 }
k <- 2
f(k)
```

```
class(k) <- "j"
f(k)
```

```
f.default <- function(x) { x + 100 }
f.l <- function(x) { x + 5 }
l <- structure(10, class = c("k","l"))
f(l)
```

```
f.j(7)
```

```
f(7)
```