Part I: *Functions* (15 points)

In *Part I* you have to develop your own functions. The code is part of the submission (in the PDF) and needs to follow the style guide of this class - Once you've completed this class you can follow your own rules, until then it is mandatory! - In the text documenting your code, you have to discuss what and how you think a particular function should be implemented. For example, outline the features which should be implemented and then how each feature can be covered in `R`. The functions you have to write are rudimental so these discussions can be short, however, make sure that it is obvious that you know what you do. If there is no documentation for an implementation (only code) the exercise is graded with 0 points.

Functions I:

Define a function which given an atomic vector `x` as argument, returns `x` after removing missing values. Do *NOT* use the following functions: `complete.cases`, `na.omit` or `na.exclude`. Missing values (for this exercise) are only defined by `NA`s. Use the following scheme:

```
dropNa <- function(x) {
  # expects an atomic vector as an argument and returns it without missing
  # values
}
```

You can use the following line to test your implementation:

```
all.equal(dropNa(c(1, 2, 3, NA, 1, 2, 3)), c(1, 2, 3, 1, 2, 3))
```

```
## [1] TRUE
```

Functions II:

Part I: Write a function `meanVarSdSe` that takes a numeric vector `x` as argument. The function should return a named *numeric* vector that contains the mean, the variance, the standard deviation and the standard error of `x`. The standard error is defined as

$$se(x) = \frac{sd(x)}{\sqrt{\#x}}$$

where $\#x$ denotes the cardinality, i.e. the number of elements contained in `x`. The code should have the following structure:

```
meanVarSdSe <- function(x){
  # ... your code ...
}
```

and the output should look like this:

```r
x <- 1:100
meanVarSdSe(x)
```

```
##       mean        var         sd         se 
##  50.500000 841.666667  29.011492   2.901149
```

You can use the functions `mean`, `var`, `sd` and `length`.

Part II: Look at the following code sequence. What result do you expect?

```r
x <- c(NA, 1:100)
meanVarSdSe(x)
```

Now run the code. Explain the result. Modify the function definition of `meanVarSdSe` with the argument `...`:

```r
meanVarSdSe <- function(x, ...) {
  # ... your code ...
}
```

so that the `na.rm = TRUE` argument can be passed optionally to the functions `mean`, `var` and `sd`. Make sure that the results after removing missing values are still correct from a statisticians point of view.

```r
meanVarSdSe(c(x, NA), na.rm = TRUE)
```

```
##       mean        var         sd         se 
##  50.500000 841.666667  29.011492   2.901149
```

Part III: Write an alternative version of `meanVarSdSe` in which you make use of the function definition `dropNa` from the above exercise. To confirm your results:

```r
meanVarSdSe(c(x, NA))
```

```
##       mean        var         sd         se 
##  50.500000 841.666667  29.011492   2.901149
```

Functions III:

Write an infix function `%or%` that behaves like the logical operator |, you are *NOT* allowed to use | though. *Hint: You only need the function `ifelse`.* You can use the following line to test your implementation:

```
c(TRUE, FALSE, TRUE, FALSE) %or% c(TRUE, TRUE, FALSE, FALSE)
```

```
## [1]  TRUE  TRUE  TRUE FALSE
```

Part II: *Scoping and related topics* (15 points)

Explain code means: First describe the generic concepts underlying the exercises. Then explain how these concepts are determining the results of the code in R. It should be possible to describe the concepts with only a few sentences.

Scoping I:

Explain the results of the three function calls.

```
x <- 5
y <- 7
f <- function() x * y
g <- function(x = 2, y = x) x * y
f()        # call 1
g()        # call 2
g(y = x)   # call 3
```

Scoping II:

Why and how does the following code work?

```
t <- matrix(1:6, ncol = 3, byrow = TRUE)
t(t)
```

```
##      [,1] [,2]
## [1,]    1    4
## [2,]    2    5
## [3,]    3    6
```

Scoping III:

Why do the results of `t(T)` and `t(t)` differ?

```
t <- function(...) matrix(...)
T <- t(1:6, ncol = 3, byrow = TRUE)
t(T)
```

```
##      [,1]
## [1,]    1
## [2,]    4
## [3,]    2
## [4,]    5
## [5,]    3
## [6,]    6
```

```
t <- function(...) matrix(...)
t <- t(1:6, ncol = 3, byrow = TRUE)
t(t)
```

```
##      [,1] [,2]
## [1,]    1    4
## [2,]    2    5
## [3,]    3    6
```

Dynamic lookup:

Explain the results of the five function calls and why the `rm` function in line 1 is important.

```
rm(list = ls(all.names = TRUE))
f <- function(x, y = x + 1) x + y
x <- 3
f(2)   # call 1
x <- 5
f(2)   # call 2

f <- function(y = x + 1) x + y
x <- 3
f(2)   # call 3
x <- 5
f(2)   # call 4
f()    # call 5
```