Question 1
(a)
(b)
(c)
Question 2
Question 3
Question 4
Question 5
Question 6
Question 7
Question 8
Question 9

Code

Stats 20, F23 – Homework 3

FIRST&LASTNAME – UID

04/24/24

Question 1

(a)

Hide

Hide

```
commute_times <- c(14,12,20,19,15,20,28,20,20,18)
mean_ct <- mean(commute_times)
sd_ct <- commute_times < mean_ct - sd(commute_times)
sd_ct_more <- commute_times >  mean_ct + sd(commute_times)

outside <- sd_ct | sd_ct_more

commute_times[outside]
```

```
## [1] 14 12 28
```

(b)

Hide

Hide

```
commute_times[!outside]
```

```
## [1] 20 19 15 20 20 20 18
```

(c)

Hide

Hide

```
length(commute_times[!outside])/length(commute_times)
```

```
## [1] 0.7
```

Question 2

Hide

Hide

```
NA & TRUE
```

```
## [1] NA
```

Hide

Hide

```
NA & FALSE
```

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

Hide

Hide

```
NA | TRUE
```

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

Hide

Hide

```
NA | FALSE
```

```
## [1] NA
```

NA and True yields NA because the code does not know whether or not NA is true. While it could be True or False, we cannot determine it so the code is waiting for NA and thereby returns an NA NA and False yields false because if we have a false, we already know no matter what the operation will yeild a false so we get a false NA and True yeilds true because once we have at a true and a NA, we know at least one value is true NA and false yeilds NA because we do not know whether a value is true or not, while it could be True or False, we cannot determine it so the code is waiting for NA and thereby returns an NA

Question 3

Hide

Hide

```
get_minimum_coins <- function(coins) {
  coin_count <- 0

  while(coins >= 25) {
    coin_count <- coin_count + 1
    coins <- coins - 25
  }
  while(coins >= 10) {
    coin_count <- coin_count + 1
    coins <- coins - 10
  }
  while(coins >= 5) {
    coin_count <- coin_count + 1
    coins <- coins - 5
  }
  while(coins > 0) {
    coin_count <- coin_count + 1
    coins <- coins - 1
  }
  coin_count
}
```

(a)

it does not work because the function is not vectorized. The function is not vectorized because the initial condition of the function is a single value with length of 1.

Hide

Hide

```
# coins <- c(1,10,15,25,100)
# get_minimum_coins(coins)
```

(b)

Hide

Hide

```
all_coins <- seq(1:99)
get_minimum_coins(99)
```

```
## [1] 9
```

Hide

Hide

```
get_minimum_coins(94)
```

```
## [1] 9
```

Hide

Hide

```
which( vapply(all_coins, get_minimum_coins, FUN.VALUE = numeric(1)) == max(vapply(a
ll_coins, get_minimum_coins, FUN.VALUE = numeric(1))))
```

```
## [1] 94 99
```

Question 4

(a)

Hide

Hide

```
y <- c("bears", "beets", "Battlestar Galactica")
x <- c(10,NA,NaN,1,2)
z <- c("hello",1,"c")


#x[!is.na(x)]


my_min <- function(x, na.rm = FALSE) {

  min_val <- x[1]

  if(na.rm == TRUE) {
    x <- x[!is.na(x)]
  }
  if(length(x) > 0) {
  for (i in 2:length(x)) {
    if (!is.na(x[i]) && x[i] < min_val) {
      min_val <- x[i]
    }
  }

  if (na.rm == FALSE && any(is.na(x))) {
    min_val <- NA
  }
} else {
  warning("no non-missing arguments to min; returning Inf")
  min_val <- Inf
}
  min_val
}

min(x, na.rm = TRUE)
```

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

Hide

Hide

```
my_min(x, na.rm = TRUE)
```

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

Hide

Hide

```
min(x, na.rm = FALSE)
```

```
## [1] NA
```

Hide

Hide

```
my_min(x, na.rm = FALSE)
```

```
## [1] NA
```

Hide

Hide

```
min(y)
```

```
## [1] "Battlestar Galactica"
```

Hide

Hide

```
my_min(y)
```

```
## [1] "Battlestar Galactica"
```

Hide

Hide

```
min(z)
```

```
## [1] "1"
```

Hide

Hide

```
my_min(z)
```

```
## [1] "1"
```

Hide

Hide

```
a <- c(4, 1, 0, 2, -3, -5, -4)
b <- c("bears", "beets", "Battlestar Galactica")
c <- 7
d <- c("Pawnee", "rules", "Eagleton", NA) #with na.rm = TRUE and na.rm = FALSE
e <- NA #with na.rm = TRUE and na.rm = FALSE

min(a)
```

```
## [1] -5
```

Hide

Hide

```
my_min(a)
```

```
## [1] -5
```

Hide

Hide

```
min(b)
```

```
## [1] "Battlestar Galactica"
```

Hide

Hide

```
my_min(b)
```

```
## [1] "Battlestar Galactica"
```

Hide

Hide

```
min(c)
```

```
## [1] 7
```

Hide

Hide

```
my_min(c)
```

```
## [1] 7
```

Hide

Hide

```
min(d, na.rm = TRUE)
```

```
## [1] "Eagleton"
```

Hide

Hide

```
my_min(d, na.rm = TRUE)
```

```
## [1] "Eagleton"
```

Hide

Hide

```
min(d, na.rm = FALSE)
```

```
## [1] NA
```

Hide

Hide

```
my_min(d, na.rm = FALSE)
```

```
## [1] NA
```

Hide

Hide

```
min(e, na.rm = TRUE)
```

```
## Warning in min(e, na.rm = TRUE): no non-missing arguments to min; returning Inf
```

```
## [1] Inf
```

Hide

Hide

```
my_min(e,na.rm = TRUE)
```

```
## Warning in my_min(e, na.rm = TRUE): no non-missing arguments to min; returning
## Inf
```

```
## [1] Inf
```

Hide

Hide

```
min(e, na.rm = FALSE)
```

```
## [1] NA
```

Hide

Hide

```
my_min(e, na.rm = FALSE)
```

```
## [1] NA
```

Question 5

(a)

Hide

Hide

```
fib1 <- 1
fib2 <- 1
full_fib <- c(fib1, fib2)
while (fib1 + fib2 < 500) {
  fib2 <- fib1 + fib2
  full_fib <- c(full_fib, fib2)
  fib1 <- fib2 - fib1
}
full_fib
```

```
##  [1]   1   1   2   3   5   8  13  21  34  55  89 144 233 377
```

(b)

Hide

Hide

```
fib1 <- 1
fib2 <- 1
full_fib <- c(1,1)
while (full_fib[length(full_fib)] +  full_fib[length(full_fib) - 1] < 500) {
  full_fib <- c(full_fib, sum(full_fib[length(full_fib)],full_fib[length(full_fib)
- 1]))
}
full_fib
```

```
##  [1]   1   1   2   3   5   8  13  21  34  55  89 144 233 377
```

(c)

Hide

Hide

```
fib1 <- 1
fib2 <- 1
full_fib <- c(1,1)
while (full_fib[length(full_fib)] +  full_fib[length(full_fib) - 1] < 10^9) {
  full_fib <- c(full_fib, sum(full_fib[length(full_fib)],full_fib[length(full_fib)
- 1]))
}
length(full_fib)
```

```
## [1] 44
```

Question 6

Hide

Hide

```
jerry <- 2:(8 * 5 %% 3^-(2:7 > 2))
```

```
## Warning in 2:(8 * 5%%3^-(2:7 > 2)): numerical expression has 6 elements: only
## the first used
```

Hide

Hide

```
jerry[1:2]
```

```
## [1] 2 1
```

Question 7

(a)

Hide

Hide

```
x <- (1:10) * pi
my_ifelse <- function(test, yes, no) {
for (i in seq_along(x)) {
  if(test[i] == TRUE){
    x[i] <- yes[i]
  }
  if(test[i] == FALSE) {
    x[i] <- no[i]
  }
}
 x
}
```

(b)

Hide

Hide

```
x <- (1:10) * pi
my_ifelse(x %% 1 >= 0.5, x %/% 1 + 1, x %/% 1)
```

```
##  [1]  3  6  9 13 16 19 22 25 28 31
```

(c)

Hide

Hide

```
my_abs <- function() {

}

my_sign <- function() {

}
```

Question 8

(a)

Hide

Hide

```
merge <- function() {

}

merge_sort <- function() {

}
```

(b)

Hide

Hide

```
# merge_sort(numeric(0))
# merge_sort(7)
# merge_sort(10:1) #cannot knit without commenting this out
```

Question 9

Hide

Hide

```
load("dna.RData")
```

(a)

(b)