HW 2

Enter your name and EID here: Harini Shanmugam

You will submit this homework assignment as a pdf file on Gradescope.

For all questions, include the R commands/functions that you used to find your answer (show R chunk). Answers without supporting code will not receive credit. Write full sentences to describe your findings.

The goal of this assignment is to encode your name (or any other message) using a *cipher* function: We want to replace each letter of a given character vector with the letter of the alphabet that is *k* positions after it in the alphabet. For example, if the letter was `a` and `k = 3`, we would replace it with `d`. We will also want it to loop around, so if the letter was `y` and `k = 3`, we'd replace it with `b`. For example, with `k = 3`, the word `dog` would become `grj`. Let's take it step by step.

Question 1: (2 pts)

Type the word `letters` into the R chunk below. What does this predefined object in R contain? What is this object's data type/class? How many elements does it contain? *Include base R commands used to answer all three questions.*

```
letters # shows the data in the object "letters"
```

```
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s"
## [20] "t" "u" "v" "w" "x" "y" "z"
```

```
summary(letters) # shows summary or length/class/mode of 'letters' object
```

```
##    Length     Class      Mode
##        26 character character
```

This predefined object contains the 26 lower-case letters of the Roman alphabet. The object's data type is characters. It contains 26 elements.

Question 2: (2 pts)

First, here is the code to split a word into a vector containing each letter.

```
test <- unlist(strsplit("test", split = ""))
# Note: the function strsplit() returns a list, use unlist() to return a vector
```

Remember that `A %in% B` returns a vector of the positions of matches of an object A in an object B (Worksheet 2):

```
letters %in% test
```

```
##  [1] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [13] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE
## [25] FALSE FALSE
```

How many elements are false in the resulting logical vector? *Include base R commands used to answer this question (recall: TRUE is equivalent to the value 1 and FALSE is 0.*

```
# Counts number of true/false in letters object
summary(letters %in% test)
```

```
##    Mode   FALSE    TRUE
## logical      23       3
```

There are 23 elements that are false.

Question 3: (2 pts)

Another function that will be useful is `which()`: it takes a logical vector as an input and returns the indices/positions that are TRUE. For example, run the following code:

```
# Note: T is shorthand for TRUE, F for FALSE
which(c(F,T,F,T,F,T))
```

```
## [1] 2 4 6
```

The output tells you that elements in position 2, 4, and 6 are true. Now, use the `which` function, along with `%in%` and `letters`, to find which positions in the alphabet the letters in the name `layla` occupy (saved as an object called `name_v` below). Would that combination of functions alone work to encode a name? Why/Why not?

```
# Define name as a vector
name_v <- unlist(strsplit("layla", split = ""))

# find which position of the alphabet letters in name_v are in
which(letters %in% name_v)
```

```
## [1]  1 12 25
```

No, this combination of functions alone would not work to encode a name because some letters may be repeated. We would not know which letters were repeated, how many times they were repeated, and in which order/positions they would be in.

Question 4: (2 pts)

How can we avoid this? For example, we can test each letter one at a time in their correct order! One approach would be to use a *for loop*. Write a for loop that goes through each element of the character vector `name_v` (i.e., each letter in `c("l","a","y","la")`) one at a time, finds its position in the alphabet, and saves each position in a vector called `positions`. Confirm that the positions are correct by using `positions` as an index to find the corresponding letters in the object `letters`.

For example, the name `ali` *would give you the positions 1,12, and 9. You can grab the letters in those positions by doing* `letters[c(1, 12, 9)]`.

```
# creating open vector for the positions object
positions <- c()
# for loop, find position of each letter of name_v
for (i in name_v){
  positions <- c(positions, which(letters %in% i))
}
positions
```

```
## [1] 12  1 25 12  1
```

Question 5: (2 pts)

Let's encode the name `layla`! Shift all the `positions` by 1 and index `letters` to obtain the encoded name. Is the encoded name a real name?

```
# create open vector for the positions object
positions <- c()
# for loop, find position of each letter of name_v
for (i in name_v){
  positions <- c(positions, which(letters %in% i))
}

letters[positions+1]
```

```
## [1] "m" "b" "z" "m" "b"
```

The output was "mbzmb" and that is not a real name.

Question 6: (2 pts)

Now, what if you would like to get the positions in your name? Or any other name? We would have to repeat questions 2-5... Instead let's write a function to 1) split a name as a vector (i.e., a character vector whose elements contain single letters), 2) initialize the positions, and 3) report each position in a vector `positions` with a for loop for each new name we would like to encode. The function should take a `name` (for example, "layla") as the input and return the alphabetical positions each of those letters occupy. Call the function `get_position`. Once you have defined it, test it out with "layla". Did you get all positions?

```r
# create new function "get_position" that can take any name as an input (input parameter = name)
get_position <- function(name){
  name_v <- unlist(strsplit(name, split = "")) # split name into list of letters, then join into one ve
  positions <- c() # create open vector for the positions object
  for (i in name_v){ # for loop, find position of each letter of name_v
  positions <- c(positions, which(letters %in% i))
  }
  return(positions)
}
get_position("layla") # can enter whichever name desired
```

```
## [1] 12  1 25 12  1
```

Yes, I got all positions.

Question 7: (2 pts)

What happens when we shift the positions past `z`, the 26th and final letter of the alphabet? Shifting the positions in `layla` up by `k = 2` should give `ncanc`, but since there is no 27th element of letters, it will return `NA` instead of `a`. Try it in the code chunk below.

```r
letters[get_position(name_v) + 2]
```

```
## [1] "n" "c" NA  "n" "c"
```

```r
# returns NA instead of a :(
```

How do we make it loop around so that `z` shifted up 1 becomes `a`? In other words, how can we make 27 become 1, 28 become 2, 29 becomes 3, etc.? We will use a mathematical operator called modulo `%%` (which tells you the remainder when you divide one number by another). Try running the code below, `27 %% 26` (pronounced "27 modulo 26"). It returns 1, the remainder when the number on the left (27) is divided by the number on the right (26).

```r
27 %% 26
```

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

We just need our shifted positions *modulo* 26. You can do this with `(positions + k) %% 26`. One last minor issue: `26 %% 26` is 0 (or any multiple of 26 %% 26 is 0) but we want it to return 26 (i.e., the letter `z`). We can fix this issue by using `ifelse` for example. Test if `positions + k %% 26` is 0: if it is, use 26, if it is not use `positions + k %% 26`. Use your function `get_position()` and the fix with modulo `%%` in `ifelse()` to encode the word `layla` by shifting every letter `k = 2` positions forward correctly. Is the encoded name a real name?

```r
# in the letters data set, find the letter corresponding to the position
# if position + shift by 2 %% 26 == 0, then still give me the letter in the 26th position which is z. O
# return z for 26 %% 26
letters[ifelse((get_position("layla") + 2) %% 26 == 0, 26, (get_position("layla") + 2) %% 26)]
```

```
## [1] "n" "c" "a" "n" "c"
```

No, the encoded name is not a real name.

Question 8: (2 pts)

Putting it all together: Write a function that incorporates all the work you have done to achieve the encoding task. Name the function `cipher`. This function should take two arguments: a `name` (a string) and how many positions to shift (`k`). Fill in the code below with what you have been using above. Check your code with `layla` shifted by 2 positions and test your code with your own name with the shift of your choice! Is the encoded name a real name?

```
# create cipher function with parameters name and k to find every letter of any name shifted by any num
cipher <- function(name, k) {
  # in the letters data set, find the letter corresponding to the position
  letters[ifelse((get_position(name) + k) %% 26 == 0, 26, (get_position(name) + k) %% 26)]
}

# check
cipher("layla", 2)
```

```
## [1] "n" "c" "a" "n" "c"
```

```
# test your name!
cipher("harini", 2)
```

```
## [1] "j" "c" "t" "k" "p" "k"
```

No, the encoded name is not a real name.

Question 9: (2 pts)

A less guided question... You were given the code `oldp`. Can you decipher the code and find the name hidden behind it?

```
# oldp shifted by every possible k/ shift value
for (k in 1:26){
  print(cipher("oldp",k))
}
```

```
## [1] "p" "m" "e" "q"
## [1] "q" "n" "f" "r"
## [1] "r" "o" "g" "s"
## [1] "s" "p" "h" "t"
## [1] "t" "q" "i" "u"
## [1] "u" "r" "j" "v"
```

```
## [1] "v" "s" "k" "w"
## [1] "w" "t" "l" "x"
## [1] "x" "u" "m" "y"
## [1] "y" "v" "n" "z"
## [1] "z" "w" "o" "a"
## [1] "a" "x" "p" "b"
## [1] "b" "y" "q" "c"
## [1] "c" "z" "r" "d"
## [1] "d" "a" "s" "e"
## [1] "e" "b" "t" "f"
## [1] "f" "c" "u" "g"
## [1] "g" "d" "v" "h"
## [1] "h" "e" "w" "i"
## [1] "i" "f" "x" "j"
## [1] "j" "g" "y" "k"
## [1] "k" "h" "z" "l"
## [1] "l" "i" "a" "m"
## [1] "m" "j" "b" "n"
## [1] "n" "k" "c" "o"
## [1] "o" "l" "d" "p"
```

The name is "Liam".

Formatting: (2 pts)

Comment your code, write full sentences, and knit your file!

```
##                                                                                                                                  sysna
##                                                                                                                                 "Darwi
##                                                                                                                                  relea
##                                                                                                                                 "21.6
##                                                                                                                                  versi
## "Darwin Kernel Version 21.6.0: Wed Aug 10 14:28:35 PDT 2022; root:xnu-8020.141.5~2/RELEASE_ARM64_T810
##                                                                                                                               nodena
##                                                                                                  "Harinis-MacBook-Air.loca
##                                                                                                                                machi
##                                                                                                                                "arm6
##                                                                                                                                  log
##                                                                                                                                 "roo
##                                                                                                                                   us
##                                                                                                                "harinishanmuga
##                                                                                                                   effective_us
##                                                                                                                "harinishanmuga
```