```
#|*****************************************************
Faisal Ibrahim                                  Homework 1
CISC 3140-ET6                                Date: 09/17/19
Prof. Murray Gross
*****************************************************|#

#|In this program we are searching for "no of 3's" from
a list integers.
This list can contain as many numbers, we just have to count,
how many 3's are there?
It can be like counter in any other programming language BUT
since this is scheme we don't have memory. It has to be just in
one flow.
I am going to use recursion for that purpose
|#

; PROGRAM CODE:
(define (count lst) ; defined the function count
                    ; which takes a list
  ; now I am running my first if statement
  ; NOTE: very important:
         #|this if statement is very important. what it does
           is it checks if the list has a null element.
           Note: when we do recursion automatically the
           last element left in the list is '() which
           represents null and we must handle this situation.
           I was trying to run the
           code without taking care of it but I was getting
           error. Going into steps I noticed the recursion
           was going fine untill the last step where my code
           had to deal with null value. The error occured
           at (car lst) in the last recursive call.
           The error reported was needed an element but
           received '()|#
  (if (null? lst) ; so this if checks if the list has come to null
                  ; since every list has a null element in it.
      0 ; so 0 if list has ended
      (if ; for the 2nd part of our we have another "if"
          ; this "if" is for our requirement which checks
          ; if the element we are getting is 3 or not?
       (= 3 (car lst)) ; = checks if "car lst" (first element
                       ; in the list) is equal to 3 or not?
       (+ 1 (count (cdr lst))) ; if our contition is met we add 1
                               ; and do recursive call on the
                               ; "cdr lst" which is
                               ; the remainder of the list
       (count (cdr lst))))) ; if out condition is not met we dont
                            ; add but just do the recursive call
                            ; on the remaining list.

"RESULTS"
; TESTS:
"1st Test"
(count (list 1 2 3 3 4 3 4 3 5 6 3 4))
                                ; we check this list where no of
                                ; 3's is 5, so we should get 5 as
"2nd Test"                      ; the result.
(count (list 1 2 4 5 6)) ; here we have no 3's so we should get 0
```

```
#| OUTPUT is:
5
0
|#

; RESULT--> Success

#|
I have another idea: we can generalize it, ie; why just search for 3,
we should be able to provide any number and look for it.
|#

"------------------------VERSION 2----------------------------------"

(define (my_count x lst) ; where x is our required number

  (if (null? lst) ; samething as before
      0
      (if (= x (car lst)) ; instead of 3 we compare car of list
                          ; with our given number x
          (+ 1 (my_count x (cdr lst))) ; same recursive call +1
          (my_count x (cdr lst)))))

; TEST FOR VER 2

; I am using the same list but with ver2
"RESULTS"
"Test 1"
(my_count 3 (list 1 2 3 3 4 3 4 3 5 6 3 4))
; answer should be same as 5
"TEST 2"
; lets try a different number this time "how many 4's".
(my_count 4 (list 1 2 3 3 4 3 4 3 5 6 3 4)); it works

; Hurraaaaaaay.....................
```