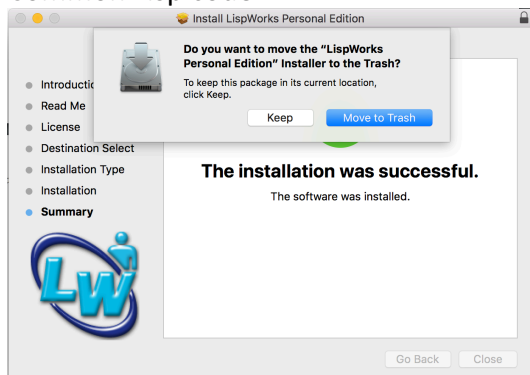


Dylan Murphy
CSI 333
Assignment 3

I decided to write my code in LISP because I found out that it's the second oldest High Level Programming Language. It first appeared in 1958. It is a beautiful language ((if)(you)(don't)(mind)(looking)(at)(parentheses)) and I just like how old it is. It is written in true prefix notation or "Polish Notation" ex:
(operator operand operand)
(+ 2 2) => 4
(myFunction myParam myParam)

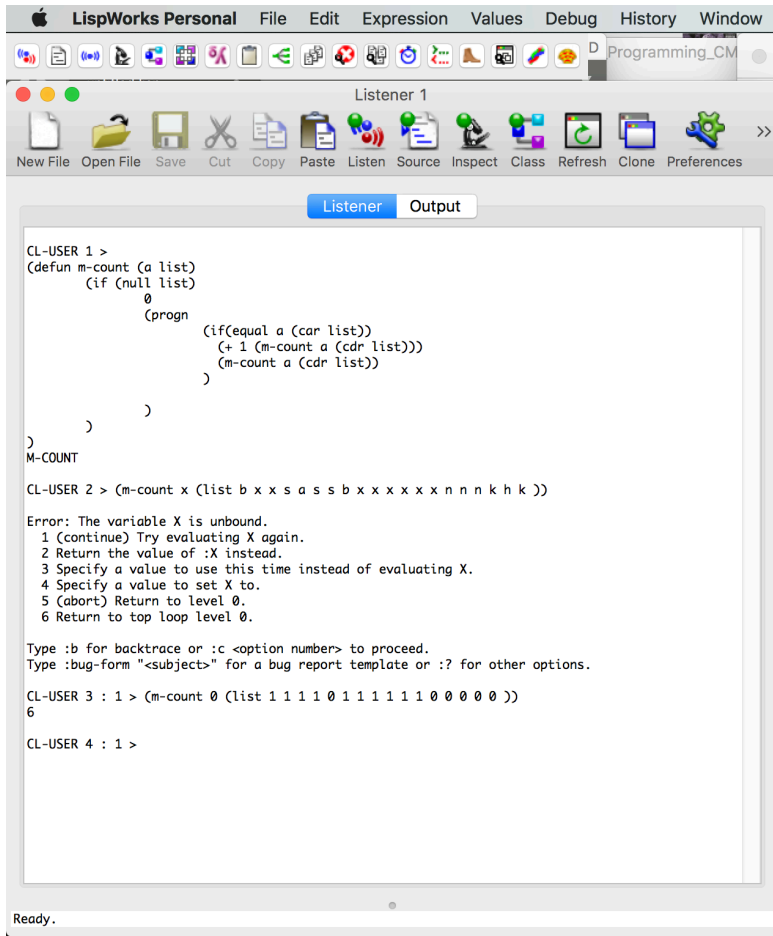
I used the application LispWorks to run my code with the necessary applications to run Common Lisp code



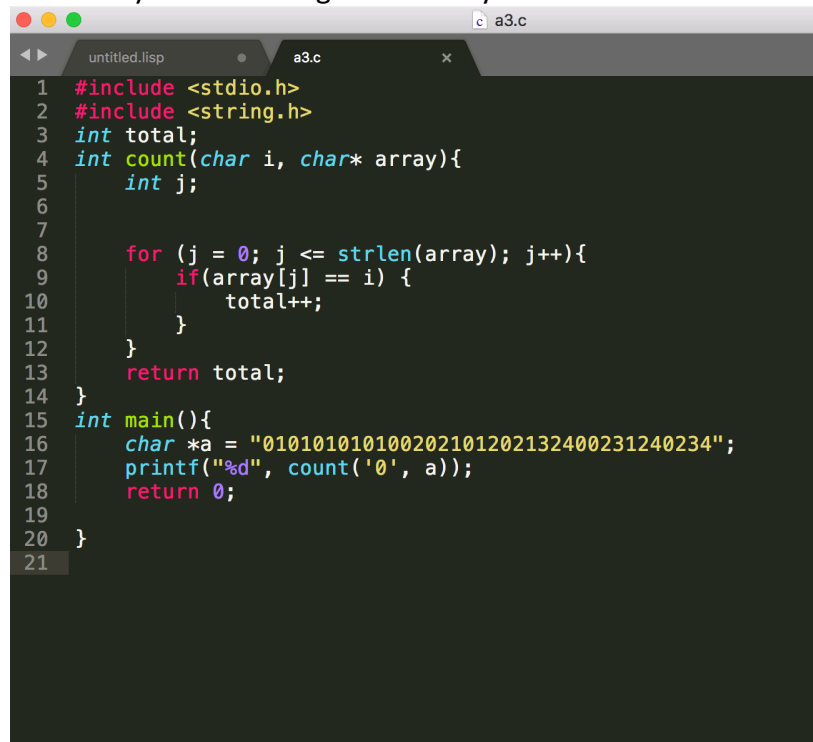
Due to my lack of experience with LISP I first took to Youtube and started absorbing the syntax structure and basic fundamentals. Once I knew what I needed to do I started coding some other basic List functions then I made my counter function. It takes an atom and counts how many times it occurs in a list. It does this recursively using LISP's super easy list splitting functions *cdr* to get the rest of the list besides the first item and *car* which returns the first item. I only got it to work for numbers because when I'm setting up the list with chars it wasn't working. I think it had something to do with how I was creating the list.

```
untitled.lisp
1  #|
2  Dylan Murphy
3  CSI 311
4  Assignment 3 Functional Program
5
6  PL: LISP
7
8  This is my first time ever writing LISP code
9  ||#
10
11  (defun m-length (list)
12    (if (null list)
13        0
14        (+ 1 (m-length (cdr list)))))
15  )
16
17
18  (defun print-list (list)
19    (if (not (null list))
20        (progn
21            (princ (car list))
22            (print-list (cdr list)))
23        )
24  )
25
26
27  (defun m-count (a list)
28    (if (null list)
29        0
30        (progn
31            (if (equal a (car list))
32                (+ 1 (m-count a (cdr list))))
33            (m-count a (cdr list))
34        )
35  )
36
37
38  )
39
40
41
42
```

I really enjoyed messing around with LISP and I really like it. Following the lambda calculus model excellently it was really easy to understand how things are structured in lisp and provides very little ambiguity. It was also very fast to debug because there was no compilation step you can just put code into the listener and let it rip!



In C I wrote a similar code to count occurrences of a char in a specific string. The C program is not doing this functionally but it is doing it iteratively.

A screenshot of a code editor window titled 'a3.c'. The code is written in C and implements a function 'count' to count the occurrences of a character 'i' in a string 'array'. The 'main' function uses this to count the number of '0's in a binary string. The code is as follows:

```
1 #include <stdio.h>
2 #include <string.h>
3 int total;
4 int count(char i, char* array){
5     int j;
6
7     for (j = 0; j <= strlen(array); j++){
8         if(array[j] == i) {
9             total++;
10        }
11    }
12    return total;
13 }
14
15 int main(){
16     char *a = "010101010100202101202132400231240234";
17     printf("%d", count('0', a));
18     return 0;
19 }
20
21
```