Scott Waldron
October 18, 2016
CSCI301 Lab4

1. Atoms can be what types of data?

    Atoms can be strings, numbers, symbols, booleans and characters.

2. What is the function definition of sub1?

    sub1 takes one arguement. It subtracts 1 from the number.

3. What is the function definition of add1? What is it similar to?

    add1 takes one arguement. It adds 1 to the number. It is similar to cons.

4. What does zero? ask? What is it similar to?

    zero? asks whether the arguement is zero. It is very similar to null except it works with numbers.

5. When recurring on a lat, what two questions should you ask?

    null? should be asked to see if the list is empty and else.

6. When recurring on a number, what two questions should you ask?

    zero? should be asked to see if the number is zero and else.

7. When recurring on an l, what three questions should you ask?

    You need to know what kind of list it is. Then you ask zero? or null?, then you ask else.

8. What is a tuple in Scheme?

    It is either an empty list, or it contains a number and a rest that is also a tuple.

9. What does addtup do?

    It builds a number by totalling all the numbers in its arguement.

10. What is the terminal condition line for lists?

```
    (null? l)
```

11. How does x use + to build numbers?

    It builds up a number by adding the first number up, the second number of times.

12. Describe the function tup+, what are its arguments?

    It adds the first number of the first tuple to the first number of the second tuple, then it adds the second number of the first tuple to the second number of the second tuple and so on. It takes two tuples
    of the same length.

13. Give the definition for the function = using zero? and sub1.

```
    (define =
        (lambda (n m)
                (cond
                        ((zero? m)(zero? n))
                        ((zero? n) #f)
                        (else (= (sub1 n) (sub1 m))))))
    It recursively sees if both m and n hit zero at the same time by subtracting one
each time.

14. Give the definition for the function = using > and <.

    (define =
        (lambda (n m)
                (cond
                        ((> n m) #f)
                        ((< n m) #f)
                        (else #t))))
    It runs the greater than and less than definitions to see if n is greater than or
less than m. If it is, it returns false.. Else, it's true.

15. Are the above functions for = the same? If so, why?

    More or less, these two functions are the same. In one case we are recursively
seeing if they reach zero together and in the other case we are seeing if n is
greater or less than m, which is the same thing.
```