PROBLEMS SHEET 1

1. Using only the **first** and **rest** functions, write the sequences of expressions that retrieve the X from the following lists/vectors. Do not attempt to do this by trial & error, work through it on paper first. Don't forget the quote character in front of the lists when you try your solutions out.

1.1: `‘(a X b c)`
1.2: `‘[a b X c]`
1.3: `‘(a b [X] c)`
1.4: `‘([X])`
1.5: `‘(a (b (X (c))))`
1.6: `‘(a (b (X)) c)`

2. Using only the following symbols: cons 'a 'b 'c () Write the expressions which build the following lists:

2.1: `(a b c)`
2.2: `(a (b c))`
2.3: `((a) b c)`

3. **count** is a function that takes a list (or vector or ...) as its argument and returns as its value the length of that list. So **count** applied to '(a b c d) gives a value of 4. Look at the lists in problem-2 and 1.5 & 1.6 in problem-1, noting what you think their lengths should be, then check using **count**. If you find the answer is not what you had expected make sure you know why.

4. Write the following functions

4.1: inc-num
takes 1 arg which is a number & returns the number incremented by 1 – keep it simple

```
eg:    (inc-num 5)  =>  6
```

4.2: inc-1st
takes 1 arg which is a list of numbers
returns the list with the first number incremented

```
eg:    (inc-1st '(1 2 3 4))  =>  (2 2 3 4)
```

4.3: vector-head
takes a list as its arg & returns the list with the first item *vectorised* (you will need to find out how to get something into a vector)

```
eg:    (vector-head '(a b c))  =>  ( [a] b c )
```

5. find the most appropriate way to retrieve the spam from the following data structures...

5.1: `'{:a egg, :b spam, :c chips}`
5.2: `'(cat dog {:a egg, :b spam, :c chips} rat frog)`
5.3:
```
'(cat [dog bat] {fruit #{mango melon pawpaw},
   breakfast {:a egg, :b spam, :c chips}} rat frog)
```
5.4: investigate at 5.3 & describe all the data structures it contains.

6. There is a number puzzle where you start from one number and have to reach another number in as few steps as possible using only a small number of mathematical operations. For example: given legal mathematical operations of multiplying by 10, dividing by 2, adding 5 and subtracting 3, what step are reuired to get from 7 to 57.

To solve this problem you need to write a transformation function which takes one number **n** as its argument and returns a list of numbers which represent all the legal transformations of **n**. So if this function was given a value of 10 it would return the list (100 5 15 3).

Write the transformation function.

7. Design a state representation and a transformation function for the "*Farmer, Dog, Goose, Corn*" problem.