PROBLEM SHEET 4.

There are some awkward problems on this sheet, you should tackle them in order as I have tried to get one to lead on to the next. You should not expect to finish this sheet in 1 hour tutorial.

4.1 Write a Fn, replace, which globally replaces some specified item in a list. Replace should take 3 args:

(a) the item to be replaced
(b) the new item
(c) the list

eg:

```
(replace 'the 'a '(the cat sat on the mat))
==>  (a cat sat on a mat)
```

4.2 If you have managed 4.1 then you have almost certainly developed a head recursive Fn. Why is tail recursion inappropriate for this type of problem. Try making the Fn tail recursive & look at its output.

4.3 Write a tail recursive reverse Fn to reverse the order of items in a list. Note CL has a Fn called "reverse" so you will have to call it something else.

4.4 APPEND is a Fn which takes 2 lists and concatenates them, ie:

```
(append '(a b c) '(d e f))  ==>  (a b c d e f)
(append '(a b c) '(x) )     ==>  (a b c x)
```

Write a Fn back-cons which takes 2 args, a list and an item & produces a new list formed by putting its 2nd arg on the back of its 1st.

ie:

```
(back-cons '(a b c) 'x)   ==>   (a b c x)
```

Do not use reverse to develop this Fn, do use append.

4.5 Write a head recursive version of reverse.

4.6 Test your reverse Fn (either head or tail version) with the following:

```
((a b c) d (e f g))
```

Alter the Fn so that with the above argument it returns:

```
((g f e) d (c b a))
```