Exercise 2.11

Work out some examples of substitution, using concrete syntax

here y is to replace all free occurrences of z

```
1. (lambda-calculus-subst
        '(lambda (x) x)
        'y                          ; exp to be subst.
        'z))
     =  (lambda (x) x)  ; { because z
                          { is not free in
                          { (lambda (x) x)

                  l-c-s
2. (lambda-calculus-subst
        '(lambda (x) x)
        'y                          { because x
        'x)                       ; { is not free in
                                    { (lambda (x) x)
     =      (lambda (x) x)

3. (l-c-s
     '(lambda (x) x)                { same
     '(+ y x)                     ; {
     'x)   =                        { reason
        (lambda (x) x)
```

```
4.  (l-c-s
      '(lambda (x) (+ x y))
      '(* x y)
      'y)   =                        ; { to avoid
                                     ;   capturing the
                                     ;   x in
                                     ;     (* x y)
    (lambda (newφ)
      (+ newφ (* x y)))


5.  (l-c-s
      '(lambda (x)
        (+ x y
           (lambda (x)
                  (* x 3))))
      '(* x y)
      'y)
              =

   (lambda (newφ)
     (+ newφ (* x y)
          (lambda (x) (* x 3))))
```

Not sure we learn anything from this !

can we make use of an 'env' parameter -

```
6.  (l-c-s
     '(lambda (x)
        (+ x y
          (lambda (y)
            (* x
              (lambda (z)
                (* x y z))))))
     '(+ x y z)
     'y)
```

env at this point would ((x))

env at this point ((y) (x))

env here ((z) (y) (x))

```
7.  (l-c-s
     '(+ (lambda (x) (+ x y))
         (lambda (y) (* x y)))
     'z
     'x )
```

Here x and y are <u>both</u> free and bound in exp

```
8.  (l-c-5
      '(lambda (x)
          (+ x y
             (lambda (x)
                (* y x)))
      '~
      '~ )
```

env:

((x) (x))