Clicker-less Questions for April 28

What is the difference between

```
        (define bobs (cons 'bob bobs))
```

and

```
        (define bob$ (cons$ 'bob bob$))
```

A. bobs is a valid list and bob$ is a valid stream
B. bobs is an infinite recursion and bob$ is a valid stream
C. bobs is a valid list and bob$ isn't worth $1
D. No difference -- both are infinite recursions

```
        (define bobs (cons 'bob bobs))
```
and
```
        (define bob$ (cons$ 'bob bob$))
```

Answer B: bobs is an infinite recursion and bob$ is a perfectly valid and delightful stream.

I want to define the stream Evens$ of even integers: 0 2 4 6 etc. What does this calculation tell you:

```
     0   2   4   6   8   10  12  14  16 ...
+ 2
---------------------------------
     2   4   6   8   10  12  14  16  18 ...
```

A. (define Evens$ (cons$ 0 (+$ 2 Evens$)))
B. (define Evens$ (cons$ 0 (cdr$ Evens$)))
C. (define Evens$ (cons$ 0 (map$ (lambda (x) (+ 2 x)) Evens$)))
D. It tells me that streams are very weird.

```
 0   2   4   6   8   10  12  14  16 ...
+ 2
---------------------------------
      2   4   6   8   10  12  14  16  18 ...
```

Answer C:

```
(define Evens$ (cons$ 0 (map$ (lambda (x) (+ 2 x)) Evens$)))
```

This one is a hint for one of the lab exercises. What is an easy way to make the stream of alternating 1 and -1: Alts$ = 1 -1 1 -1 1 -1 ....?

A. If you square every element you get the stream of 1s:
   One$ = 1 1 1 ...
B. If you add Alts$ to (cdr$ Alts$) you get the stream of 0s:
   Zero$ = 0 0 0 0 ... = (cons$ 0 Zero$)
C. If you multiply Alts$ by -1 and cons$ 1 onto the front you get Alts$ back.
D. (define Alts$ (cons$ 1 (cons$ -1 Alts$)))

Alts$ = 1 -1 1 -1 1 -1 ....

Answer D: (define Alts$ (cons$ 1 (cons$ -1 Alts$)))

Answer C also works:

```
(define Alts$  (cons$ 1 (map$ (lambda (x) (* -1 x)) Alts$)))
```