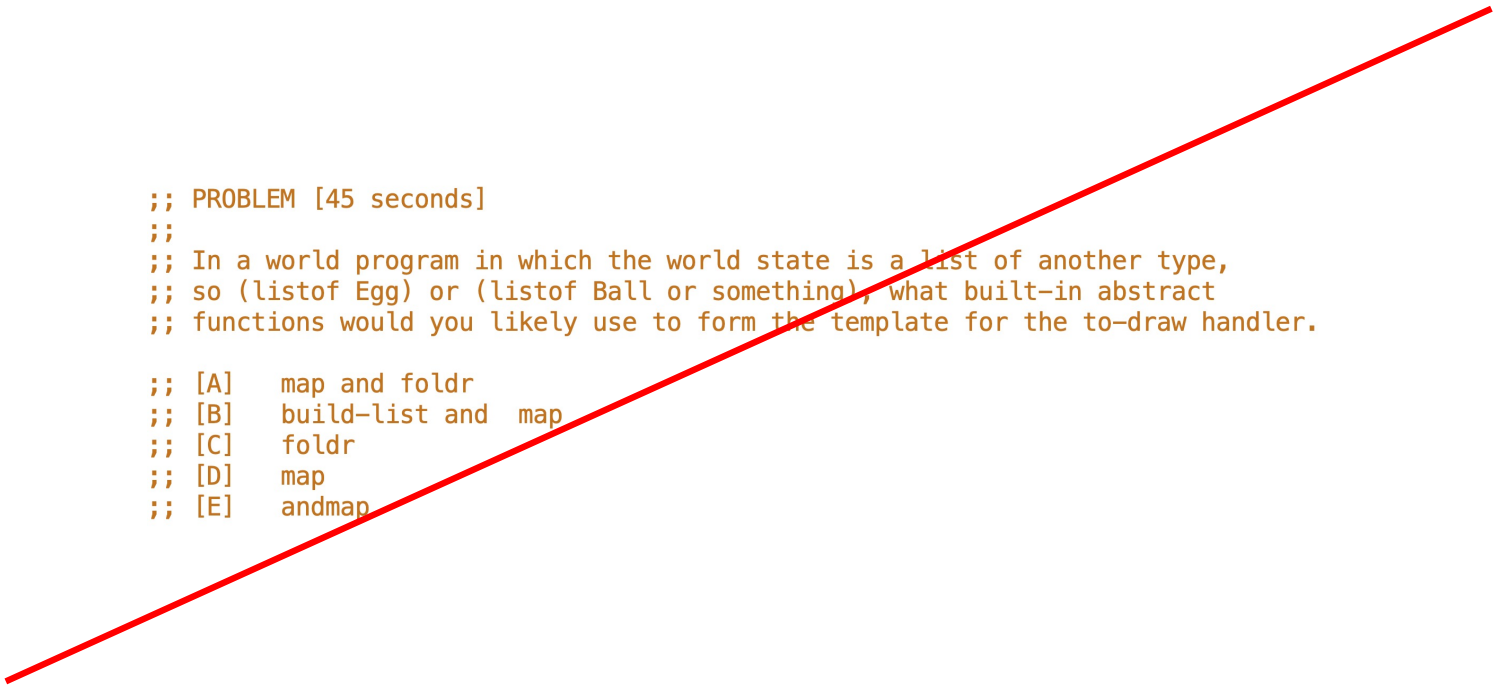
```
;; PROBLEM [45 seconds]
;;
;; In a world program in which the world state is a list of a non-primitive
;; type, so (listof Egg) or (listof Ball) or something like that, which option
;; below would be the most appropriate on-tick handler tag and template using
;; built in abstract functions?  (Just consider these choices.)
#|

;; [A]
(@template-origin function-composition use-abstract-fn)
(define (tock lox)
  (filter ... (map ... lox)))

;; [B]
(@template-origin use-abstract-fn)
(define (tock lox)
  (foldr ... lox))

;; [C]
(@template-origin use-abstract-fn)
(define (tock lox)
  (filter ... lox))

;; [D]
(@template-origin function-composition use-abstract-fn)
(define (tock lox)
  (map ... (filter ... lox)))

|#
```

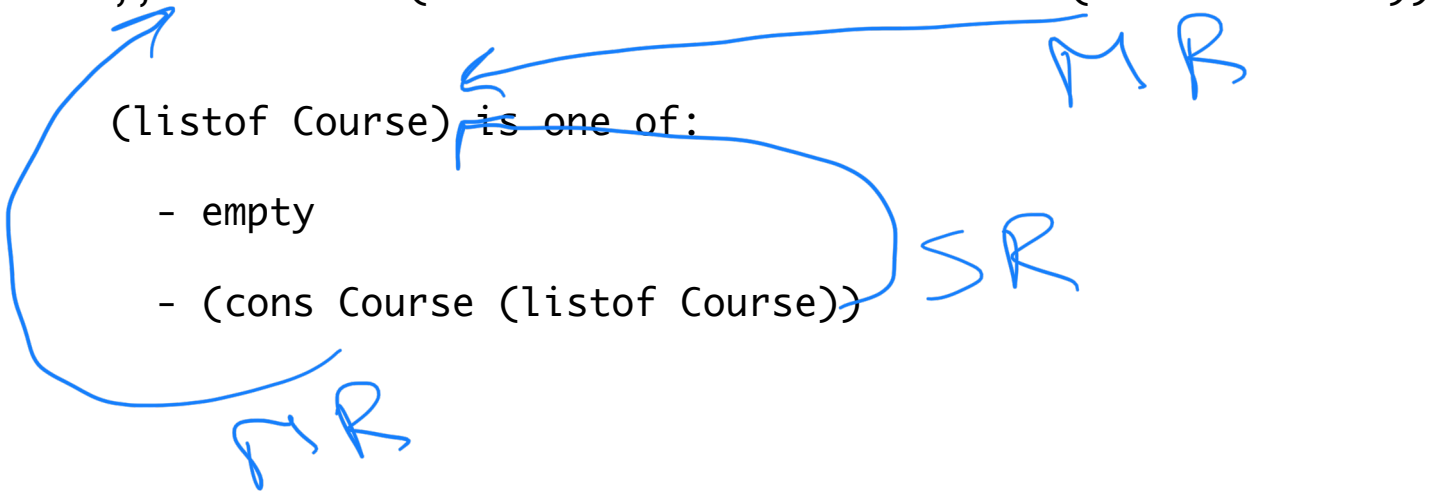
```
;; PROBLEM [45 seconds]
;;
;; In a world program in which the world state is a list of another type,
;; so (listof Egg) or (listof Ball or something), what built-in abstract
;; functions would you likely use to form the template for the to-draw handler.

;; [A]   map and foldr
;; [B]   build-list and  map
;; [C]   foldr
;; [D]   map
;; [E]   andmap
```

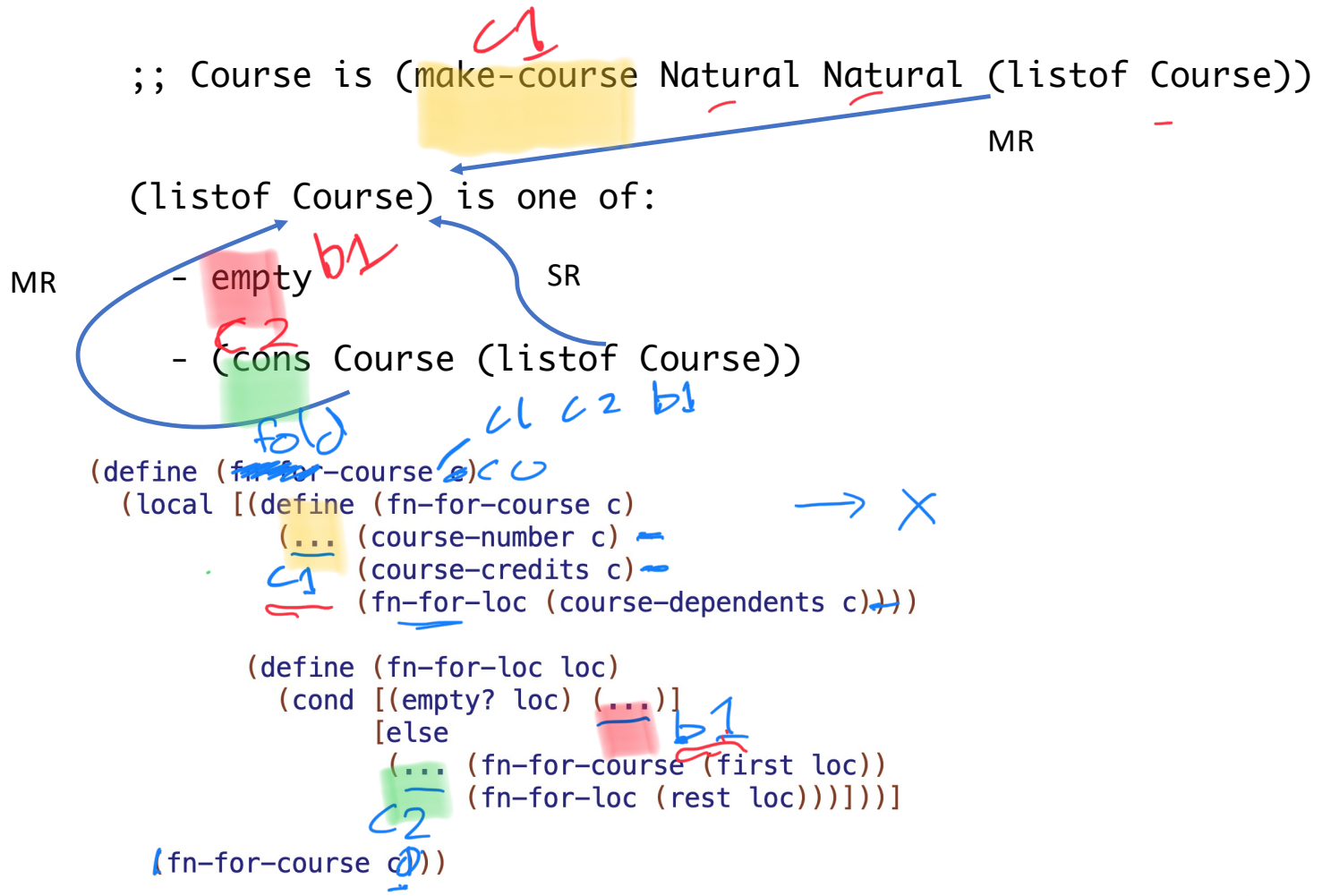


```
;; PROBLEM
;;
;; Given the following function definition:
;;

(define (add-all a b c lon)
  (local [(define d (+ b c))
          (define (adder n)
            (+ n a d))]
    (map adder lon)))

(add-all 1 2 3 (list 6 7 8))

;; What does (add-all 1 2 3 (list 6 7 8)) produce? [60 seconds]
;;
;;  [A] (list 7 8 9)
;;  [B] (list 11 12 13)
;;  [C] (list 12 13 14)
;;  [D] causes an error
```

```
;; PROBLEM
;;
;; Given the following function definition:
;;

(define (add-all a b c lon)
  (local [(define d (+ b c))
          (define (adder n)
            (+ n a d))]
    (map adder lon)))

(add-all 1 2 3 (list 6 7 8))

;; What does (add-all 1 2 3 (list 6 7 8)) produce? [60 seconds]
;;
;;  [A] (list 7 8 9)
;;  [B] (list 11 12 13)
;;  [C] (list 12 13 14)
;;  [D] causes an error


;; What is the lifted defnition of adder? [45 seconds]
;;
;; [A] (define (adder_0 n) (+ n a d))
;; [B] (define (adder_0 n) (+ n 1 d))
;; [C] (define (adder_0 n) (+ n a d_0))
;; [D] (define (adder_0 n) (+ n 1 d_0))
;; [E] (define (adder_0 n) (+ n 1 5))
```

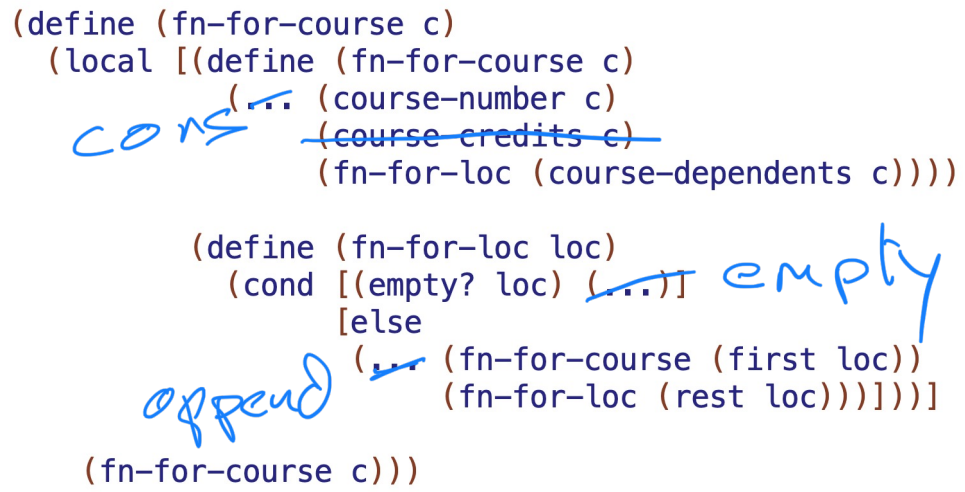
```
;; Course is (make-course Natural Natural (listof Course))

(listof Course) is one of:
  - empty
  - (cons Course (listof Course))
```



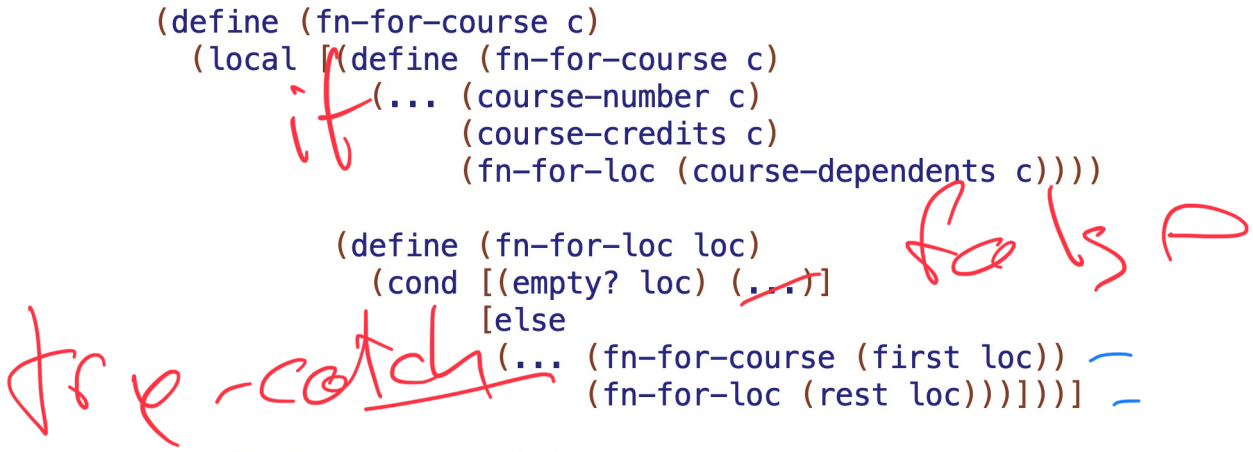

;; Course is (make-course Natural Natural (listof Course))
MR
(listof Course) is one of:
MR
- empty
SR
- (cons Course (listof Course))
(define (fn-for-course c)
(local [(define (fn-for-course c)
(... (course-number c)
(course-credits c)
(fn-for-loc (course-dependents c))))
(define (fn-for-loc loc)
(cond [(empty? loc) (...)]
[else
(... (fn-for-course (first loc))
(fn-for-loc (rest loc)))]))]
(fn-for-course c)))

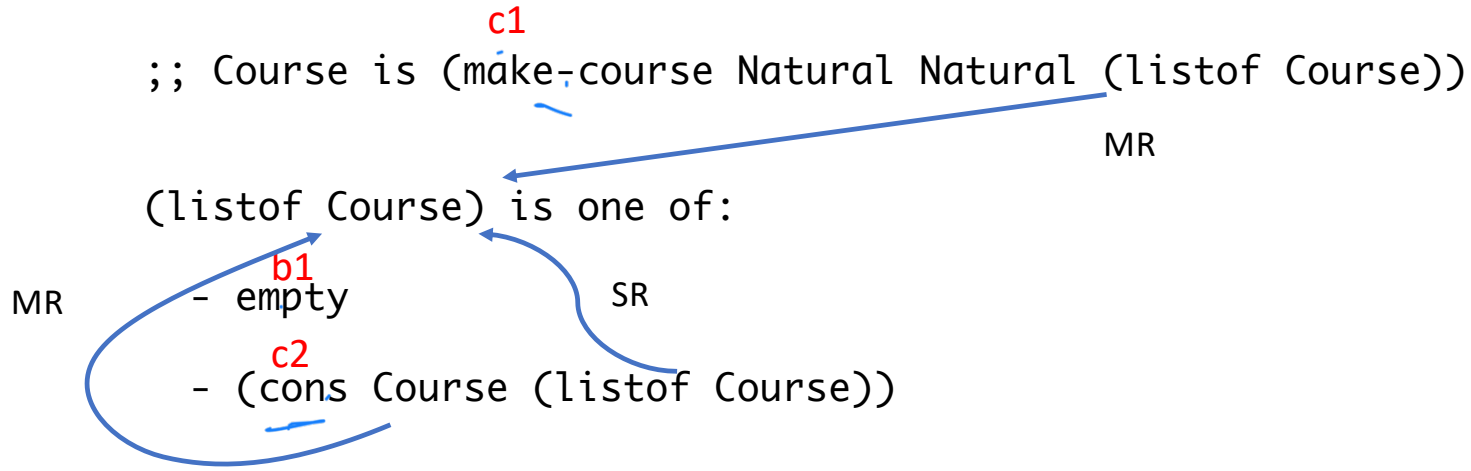
if I was doing templates

```
(define (fn-for-course c)
  (local [(define (fn-for-course c)
            (... (course-number c)
                 (course-credits c)
                 (fn-for-loc (course-dependents c))))

          (define (fn-for-loc loc)
            (cond [(empty? loc) (...)]
                  [else
                   (... (fn-for-course (first loc))
                        (fn-for-loc (rest loc)))]))]

    (fn-for-course c)))
```


(define (fn-for-course c)
(local [(define (fn-for-course c)
(... (course-number c)
(course-credits c)
(fn-for-loc (course-dependents c))))
(define (fn-for-loc loc)
(cond [(empty? loc) (...)]
[else
(... (fn-for-course (first loc))
(fn-for-loc (rest loc)))]))]
(fn-for-course c)))
cons
empty
append

```
(define (fn-for-course c)
  (local [(define (fn-for-course c)
            (... (course-number c)
                 (course-credits c)
                 (fn-for-loc (course-dependents c))))

          (define (fn-for-loc loc)
            (cond [(empty? loc) (...)]
                  [else
                   (... (fn-for-course (first loc))
                        (fn-for-loc (rest loc)))]))]

    (fn-for-course c)))
```

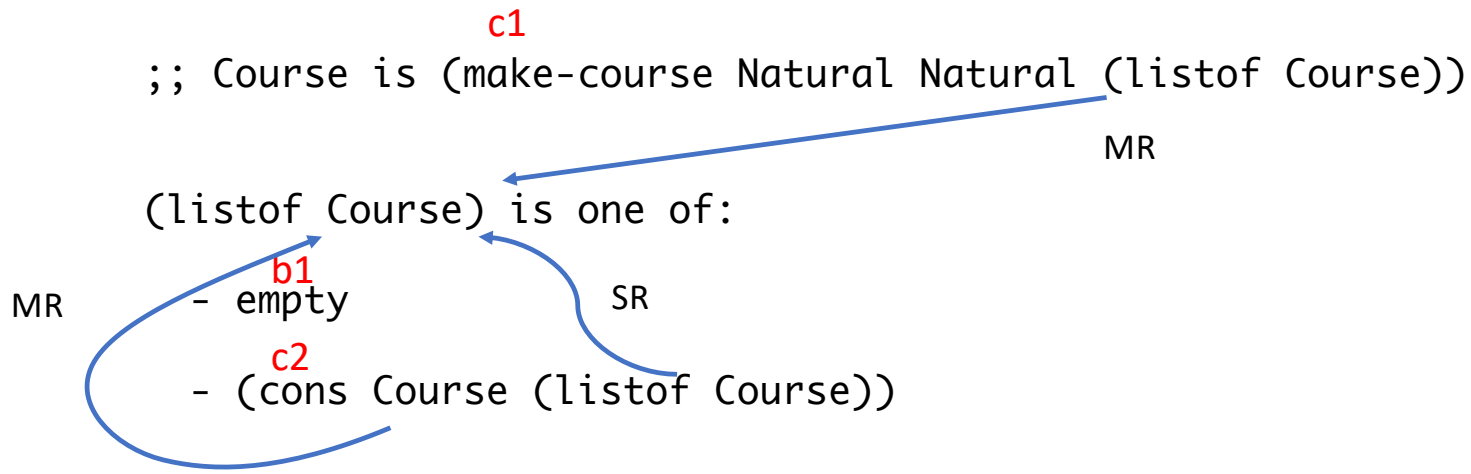




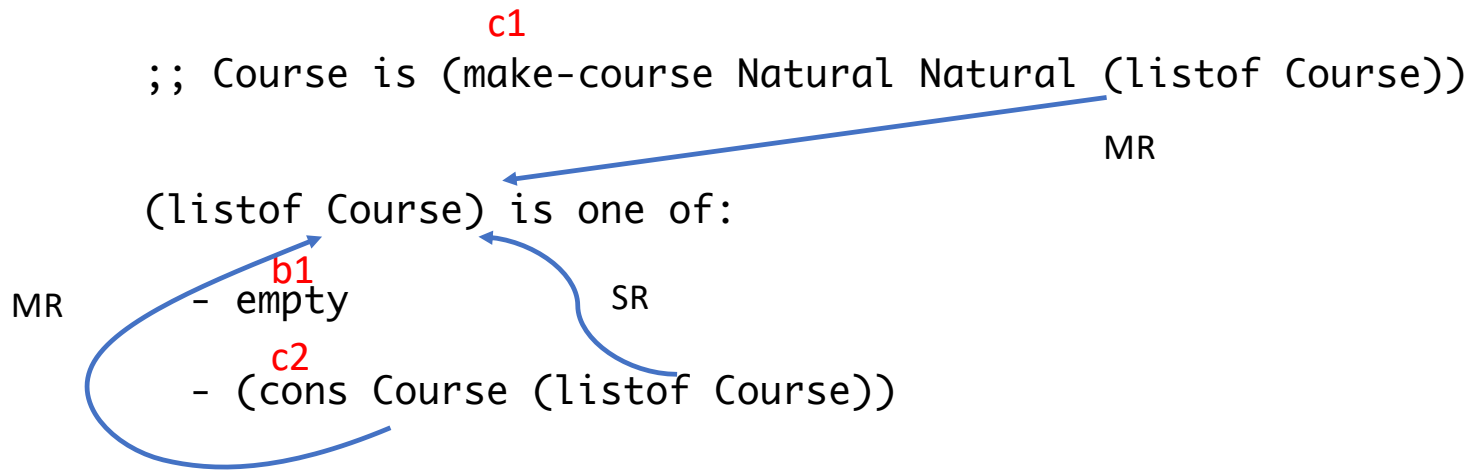
C1	C2	B1
all-numbers	(cons num rnr)	append	empty
total-credits	(+ cre rnr)	+	0
courses-w-credits	(if (≥ cre x) (cons ○ rnr) rnr)	append	empty

doesn't produce list of nice elements

can't stop early



	C1	C2	B1
All nums	(cons num rmr)	append	empty
Total credits	(+ cr rmr)	+	0



	C1	C2	B1
All nums	(cons num rmr)	append	empty
Total credits	(+ cr rmr)	+	0
Courses w/ credits	(if (>= cr n) (cons <C> rmr) rmr)	append	empty
find			

impossible

undesirable

copy template

copy template