CASSIGNMENT ON LISTS

E130 (Cont/predicate pred loci) returns a last of clamante Sorted by the predicate > (sort/predicate < ^ (8 2 5 2 3)) (22358)

> (sort/predicate > (82523))

(855322)

Coletine Sort/predicate Clambda (pred loi)

(sort loi) reforms a last of the elements of lois in ascending

E128 (merge love love) where love and low are lists of integers that are sorted in ascending order, returns a sorted lest of all the integers in lois and lois

merge ((1 4) (1 2 8))

> (merge (35 62 81 90 91) (3 83 85 90)) 35 62 81 83 82

E1.27 (flotten skirt) returns a list of symbols contained in skirt in the order in which they occur when skirt is printed.

intuitively, flatten removes all the parentheses from aix argument.

> (flatten '(a b c))

(a b c)

> (flatten '((a) () (b c)) () (c)))

(a b c)

> (flotten *((a b) c (((a)) e)))

(ab c d e)

Xflatten '(a b (() (c))))

element of let. if a top-level element is not a list, it is included in the result, as is. the value of (op (docon last)) is equilibrated to let, but (down (op let)) is not necessity.

> (up '((12) (34)))
(1234)
> (up '((x(y)) z))
(x(y) z)

(a b c)

E1.25	Cexists? pred let) returns #12 if any element of let satisfie
	pred, and returns #6 atherwise.
	> (exists? number? (abc3e))
	> (exists? number? '(a b c d e))
	#6.
£1.24	(every? pred lot) returns #f if any element of lot fails to satisfy pred, returns #t otherwise.
	> (every? number? (a.b.c.3e)) #6
	> (every? number? (1 2 3 4 5)) #+
E1.23	Clast-index gred lot returns the o-based position of the
1	first element of list that satisfies the predicate pred. if no element of the list satisfies the predicate then list-index returns #6.
	r (list-index number? (a 2 (13) b 7))
	> (list-index symbol? "(a (b c) 17 600))
	> (list-index gymbol? (1 2 (a b) · 3))

#6

E1.22 (filter-in pred lst) returns the list of those elements in Lot that satisfy the predicate pred.

> (filter-in number? 1(a 2 (1 3) b 7))
(2 7)

> (filter-in symbol? (a Cb c) 17 foo))
Ca foo)

21021 351

E1.21 (product sos1 sos2), whome sos1 and sos2 are each a list of symbols cuithout repetitions, returns a list of 2-lists that represents the Cartesian product of sos1 & sos2.

the 2-lists way may appear in any order.

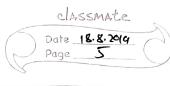
> (product '(a b c) '(x y))
((a x) (a y) (b x) (b y) (c x) (c y))

E1.20 (count-occurrences solist) returns the number of occurrences of s in slist.

> (count-occurrences 1/2 (((x x) y (((x z) x))))

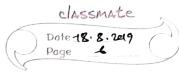
> Count-occurrences 'x '((f x) y (((x z) () x)))

Occount-occurrences w'((fx) y(((xz)(x))))



Clist-sot lot n x) returns a list like let, except that the n-th element, using zero-based indexing, is a (list-set (a b c d) 2 '(i 2)) (ab (12) d) > Clist-satref (list-set (a b c d) 3 (1 5 10)) 3) (i 5 10) E 1.18 (swapper of sz slist) returns a list, the same as slist but with all occurrences of si replaced by se and all occurrences of 52 replaced by si > (swapper 'a 'd '(a b c d)) (dbca) > (swapper a /d /(a d c) c d) (da () c a) > (swapper 1x 'y '((x) y (2 (x)))) ((y) n (z (y))) E1.17 (down lot) wrops parentheses around each top-level element > (down '(, 2 3)) ((1) (2) (3)) xdown ((a) (fine) (idea))) (c(a)) (Cfine)) (Cidea))) > (clown '(a (more (complicated)) object))

((a) ((more (complicated))) lobject))\$



£1.16	(invert let), where let is a list of 2-lists (lists of length 2) returns a list with each 2-list revensed.
	> (invert '((a1) (a2) (ib) (2b))) ((1a) (2b) (b1) (b2))
EI-18	(duple n r) retrorns a list containing n capies of re
	> (duple 2 3) (3 3) > (duple 4 (ha ha))
	(Cha ha) (ha ha) (ha ha)) > Coluple o (blan)) ()