

```
(define flatten (lambda (L)
  (cond ((null? L) nil)
        ((symbol? L) (list L))
        (#t (append
              (flatten (car L))
              (flatten (cdr L)) ) ) ) ) )
```

```
(define collapse (lambda (L)
  (cond ((null? L) nil)
        ((symbol? L) (list L))
        (#t (union
              (collapse (car L))
              (collapse (cdr L)) ) ) ) ) )
```

```
(flatten '((a b) c c ((d)) (d (g (h)))))
(A B C C D D G H)
```

```
(collapse '((a b) c c ((d)) (d (g (h)))))
(A B C D G H)
```

```
(define union (lambda (x y)
  (cond ((null? x) y)
        ((member (car x) y)
         (union (cdr x) y) )
        ((cons (car x)
                 (union (cdr x) y) ) ) ) ) ) )
```