

Computer Science I

Professor Tom Ellman
Lecture 9

Do two nested lists have the same shape?

```
(define same-shape (lambda (x y) ...?...))
```

Welcome to DrScheme.

```
> (same-shape? '(a (b) c) '(a b c))  
#f  
> (same-shape? '(a (b) c) '(a (b) c))  
#t  
> (same-shape? '(a (b) c) '(q (r) d))  
#f
```

Do two nested lists have the same shape?

```
(define same-shape?  
  (lambda (x y)  
    (or (and (null? x) (null? y))  
        (and (not (null? x)) (not (null? y))  
              (not (pair? x)) (not (pair? y)))  
        (and (pair? x)  
              (pair? y)  
              (same-shape? (car x) (car y))  
              (same-shape? (cdr x) (cdr y))))))
```

Match Corresponding Items on a Nested List

```
(define same-shape (lambda (x y) ...?...))
```

Welcome to DrScheme.

```
> (match-all* 'a 'b)  
(a b)  
> (match-all* '(a b c) '(d e f))  
((a d) (b e) (c f))  
> (match-all* '(a (b) c) '(d (e) f))  
((a d) ((b e) (c f)))
```

Match Corresponding Items on a Nested List

```
(define match-all*  
  (lambda (x y)  
    (cond ((null? x) '())  
          ((not (pair? x)) (list x y))  
          (else (cons (match-all* (car x) (car y))  
                       (match-all* (cdr x) (cdr y))))))
```

Association Lists

```
(define phone-numbers  
  '( (nancy 5988)  
      (chris 5992)  
      (tom 5991)  
      (brad 7497)  
      (lou 7293) ) )
```

Association Lists

```
(define association-list
  '( (<key> <value>)
    (<key> <value>)
    ...
    (<key> <value> ) ) )
```

Find a Value on an Association List

```
(define lookup (lambda (key alist) ...?...))
```

Welcome to DrScheme.

```
> (lookup 'b '((a 1)(b 2)(c 3)))
```

```
2
```

```
> (lookup 'd '((a 1)(b 2)(c 3)))
```

```
fail
```

```
> (lookup 'a '())
```

```
fail
```

Find a Value on an Association List

```
(define lookup
  (lambda (key alist)
    (cond ((null? alist) 'fail)
          ((equal? key (caar alist)) (cadar alist))
          (else (lookup key (cdr alist))))))
```

Put a Value on an Association List

```
(define insert (lambda (value key alist) ...?...))
```

Welcome to DrScheme.

```
> (insert 2 'b '((a 1)(c 3)))
```

```
((a 1) (c 3) (b 2))
```

```
> (insert 42 'b '((a 1)(b 2)(c 3)))
```

```
((a 1) (b 42) (c 3))
```

Put a Value on an Association List

```
(define insert
  (lambda (value key alist)
    (cond ((null? alist) (list(list key value)))
          ((equal? key (caar alist)) (cons (list key value)
                                             (cdr alist)))
          (else (cons (car alist)
                      (insert value key (cdr alist))))))
```

Remove a Value from an Association List

```
(define delete (lambda (key alist) ...?...))
```

Welcome to DrScheme.

```
> (delete 'b '((a 1)(b 2)(c 3)))
```

```
((a 1) (c 3))
```

```
> (delete 'd '((a 1)(b 2)(c 3)))
```

```
((a 1) (b 2) (c 3))
```

Remove a Value from an Association List

```
(define delete
  (lambda (key alist)
    (cond ((null? alist) '())
          ((equal? key (caar alist)) (cdr alist))
          (else (cons (car alist) (delete key (cdr alist)))))))
```