

Computer Science I

Professor Tom Ellman
Lecture 12

Are the four numbers a, b, c & d equally spaced?

```
(define equally-spaced? (lambda (a b c d) ...?...))
```

```
Welcome to DrScheme.  
> (equally-spaced? 2 4 6 8)  
#t  
> (equally-spaced? 7 7 7 7)  
#f  
> (equally-spaced? 1 3 6 8)  
#f
```

Are the four numbers a, b, c & d equally spaced?

```
(define equally-spaced-items?  
  (lambda (a b c d)  
    (and (= (- b a) (- c b)) (= (- c b) (- d c)))))
```

Given a Person's Age, Determine the Level of His/Her School

0 ... 2	None
3 ... 4	Pre-School
5	Kindergarden
6 ... 11	Elementary
12 ... 14	Middle-School
15 ... 17	High-School
18 ... 21	College
22 ...	None

Give an Person's Age, Determine the Level of His/Her School

```
(define school  
  (lambda (x)  
    (cond ((<= x 2) 'none)  
          ((<= x 4) 'pre-school)  
          ((= x 5) 'kindergarden)  
          ((<= x 11) 'elementary-school)  
          ((<= x 14) 'middle-school)  
          ((<= x 17) 'high-school)  
          ((<= x 21) 'college)  
          (else 'none))))
```

Sum of Integers from Lower Bound to Upper Bound

```
(define interval-sum (lambda (low high) ...?...))
```

```
Welcome to DrScheme.  
> (interval-sum 1 10)  
55  
> (interval-sum 1 3)  
6  
> (interval-sum 7 7)  
7
```

Sum of Integers from Lower Bound to Upper Bound

```
(define interval-sum
  (lambda (low high)
    (if (= low high)
        low
        (+ low (interval-sum (+ low 1) high))))))
```

Are the numbers on a list uniformly spaced?

```
(define uniformly-spaced? (lambda (lst) ...?...))
```

```
Welcome to DrScheme.
> (uniformly-spaced? '(2 4 6 8 10 12 14))
#t
> (uniformly-spaced? '(17 17 17 17 17 17))
#f
> (uniformly-spaced? '(7))
#t
> (uniformly-spaced? '())
#f
> (uniformly-spaced? '(2 4 6 9 11 13 15))
#f
```

Are the numbers on a list uniformly spaced?

```
(define uniformly-spaced-list?
  (lambda (lst)
    (or (null? lst)
        (null? (cdr lst))
        (null? (caddr lst))
        (and (= (- (cadr lst) (car lst)) (- (caddr lst) (cadr lst)))
              (uniformly-spaced-list? (cdr lst)))))
```

Do Corresponding Integers on Two Lists of Equal Length Have the Same (even/odd) Parity?

```
(define each-same-parity? (lambda (lst1 lst2) ...?...))
```

```
Welcome to DrScheme.
> (each-same-parity? '(2 3 7 1 4 8)
                     '(6 5 7 9 2 0))
#t
> (each-same-parity? '(2 3 7 1 4 8)
                     '(6 2 7 9 2 0))
#f
> (each-same-parity? '() '())
#t
```

Do Corresponding Integers on Two Lists of Equal Length Have the Same (even/odd) Parity?

```
(define each-same-parity?
  (lambda (lst1 lst2)
    (or (null? lst1)
        (and (same-parity? (car lst1) (car lst2))
              (each-same-parity? (cdr lst1) (cdr lst2)))))

(define same-parity?
  (lambda (x y)
    (or (and (even? x) (even? y)) (and (odd? x) (odd? y)))))
```

Mark All Items on a List to “BUY”

```
(define buy-all (lambda (lst) ...?...))
```

```
Welcome to DrScheme.
> (buy-all '(microsoft intel sun apple))
((buy microsoft) (buy intel) (buy sun) (buy apple))
> (buy-all '(ibm))
((buy ibm))
> (buy-all '())
()
```

Mark All Items on a List to “BUY”

```
(define buy-all
  (lambda (lst)
    (if (null? lst)
        '()
        (cons (buy (car lst)) (buy-all (cdr lst))))))

(define buy (lambda (x) (list 'buy x)))
```

Negate all the Numbers on a Nested List of Numbers

```
(define negate-all* (lambda (item) ...?...))
```

```
Welcome to DrScheme.
> (negate-all* '(4 (-2 3) 6))
(-4 (2 -3) -6)
> (negate-all* '())
()
> (negate-all* 7)
-7
```

Negate all the Numbers on a Nested List of Numbers

```
(define negate-all*
  (lambda (item)
    (cond ((null? item) '())
          ((number? item) (- item))
          (else (cons (negate-all* (car item))
                      (negate-all* (cdr item)))))))
```

Determine the Behavior of this Mystery Procedure

```
(define mystery
  (lambda (lst)
    (if (null? lst)
        '()
        (cons (car lst) (cons (car lst) (mystery (cdr lst)))))))
```

Determine the Behavior of this Enigmatic Procedure

```
(define enigma
  (lambda (lst n)
    (if (null? lst)
        '()
        (cons (replicate (car lst) n) (enigma (cdr lst) (+ n 1))))))

(define replicate
  (lambda (thing n)
    (if (= n 0)
        '()
        (cons thing (replicate thing (- n 1))))))
```