MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Department of Electrical Engineering and Computer Science
6.001 Structure and Interpretation of Computer Programs
Spring, 2007

Recitation 8, Wed March 7

Higher Order Procedure Notes Dr. Kimberle Koile

Very useful hops:

```
(define (map proc items)
  (if (null?  items)
      '()
      (cons (proc (car items))
            (map proc (cdr items)))))
```

```
(define (filter pred items)
   (cond ((null? items) '())
         ((pred (car items)) (cons (car items) (filter pred (cdr items))))
         (else (filter pred (cdr items)))))
```

Examples

Assume that a point is represented as a 2-element list (x y), and that you have procedures called get-x and get-y that when given a point return the x and y elements, respectively.

```
 (define make-point list)
 (define get-x car)
 (define get-y cadr)
```

1. Write a procedure get-x-coords that produces a list of all the x coordinates is a list of points.

2. Write a procedure get-greater-evens that produces a list of all the x coordinates that are even and greater than a specified number.

```
(define (fold-right op init items)
   (if (null? items)
       init
       (op (car items)
           (fold-right op init (cdr items)))))

(define (fold-left op init items)
    (if (null items)
        init
       (fold-left op (op init (car items)) (cdr items))))
```

Compare fold-right and fold-left:

```
(fold-right + 0 (list 1 2 3)) => (+  1 (+  2  (+ 3  0))) => 6
(fold-left + 0 (list 1 2 3)) => (+  (+  (+ 0  1)  2)  3) => 6

(fold-right / 1  (list 2 3 4)) => (/  2  (/  3  (/  4   1))) => 2  2/3
(fold-left / 1  (list 2 3 4)) => (/  (/  (/ 1  2)   3)   4) => 1/24
```