```
(define L '(1 2 31 4 5 66))


(define list-sum
    (lambda (L)
```

```
(list-sum '(2 -1 4))
(list-sum L)
```

```
(define L '(1 2 31 4 5 66))


(define list-sum
    (lambda (L)
      (if (null? L)
          0
          (+ (car L) (list-sum (cdr L))))))



(list-sum '(2 -1 4))
(list-sum L)
```

```
(define L '(1 2 31 4 5 66))
```

```
(define member?
  (lambda (x list)
```

```
(member? 4 L)
```

```
(define L '(1 2 31 4 5 66))
```

```
(define member?
  (lambda (x list)
     (if (null? list)
         #f
         (if (= x (car list))
             #t
             (contains? x (cdr list))))))
```

```
(member? 4 L)
```

```
(define L '(1 2 31 4 5 66))
```

```
(define member?
  (lambda (x list)
    (cond
      ((null? list) #f)
      ((= x (car list)) #t)
      (else (member? x (cdr list))))))
```

```
(member? 4 L)
```

```
(define L '(1 2 31 4 5 66))
```

```
(define max-list
  (lambda (lst)
```

```
(max-list L)
```

```
(define L '(1 2 31 4 5 66))
```

```
(define max-list
  (lambda (lst)
    (cond
      ((null? lst) (error "Empty list"))
      ((= (length lst) 1) (car lst))
      (else (max (car lst) (max-list (cdr lst)))))))
```

```
(max-list L)
```