Problem 1:

Part A:

```
(struct: a-program
        (struct: if-exp
                (struct: zero?-exp
                        (struct: var-exp: x))
                (struct: const-exp 44)
                (struct: diff-exp
                        (struct: const-exp 44)
                        (struct: var-exp: x))))
```

Part B:

(Let x = 97 in -(32, x)))

Problem 2:

```
(define (empty-list)
  (lambda (mode)
    (display "end of list")))
(define (prepend-list a lst)
  (lambda (mode)
    (if mode
      a
      lst)))
(define (car-list lst)
(lst #t))
(define (cdr-list lst)
(lst #f))
```

Problem 3:

```
(define (remove-n-times elm lst num) (cond ((= num 0) lst)
                                           ((> num (count-num-occur elm lst
0)) (remove-all-occur elm lst '()) )
                                           (else (remove-n-times elm
(remove-occur elm lst '()) (- num 1))) ))
```

```
(define (remove-occur elm lst lst-new) (if (eq? elm (car lst))
                                    (cons lst-new (cdr lst))
```

```
                                        (remove-occur elm (cdr lst)
(append lst-new (car lst)))))
```

```
(define (remove-all-occur elm lst lst-new) (cond ((null? lst) lst-new)
                                                 ((not (eq? elm (car
lst)))
                                                  (remove-all-occur elm
(cdr lst) (append lst-new (car lst))) )
                                                 (else
(remove-all-occur elm (cdr lst) lst-new) ) ))
```

```
(define (count-num-occur elm lst count) (cond ((null? lst) count)
                                               ((not (eq? elm (car lst)
))
                                                (count-num-occur elm
(cdr lst) count))
                                               (else (count-num-occur
elm (cdr lst) (+ count 1))) ))
```

Problem 4:

```
(define (count-occurrence-nested lst value) (con-helper lst value 0))
(define (con-helper lst value n) (cond ((null? lst) n)
                                       ((list? (car lst)) (+ (con-helper
(car lst) value 0) (con-helper (cdr lst) value n)))
                                       ((eq? (car lst) value) (con-helper
(cdr lst) value (+ n 1)))
                                       (else (con-helper (cdr lst) value
n))))
```