Problem 1:

Recursive:

- Has pending operations
- Time complexity is the same as the iterative
- Space complexity is worse since it has pending operations
- It requires a base case to stop the program from infinitely repeating.

Iterative:

- Doesnt have pending operations
- Time complexity is the same as the recursive
- Space complexity is better since no pending operations
- It requires an extra parameter
- It requires a base case to stop the program from infinitely repeating.

Problem 2a:

```
(define (even-odd li) (cond ((null? li) 0)
                            ((= (modulo (car li) 2) 0)
                             (+ (even-odd (cdr li)) (car li)) )
                            (else (- (even-odd (cdr li)) (car li)) ) ) )
```

Problem 2b:

```
(define (even-odd-iter li) (even-odd-help li 0))

(define (even-odd-help li x) (cond ((null? li) x)
                            ((= (modulo (car li) 2) 0)
                             (even-odd-help (cdr li) (+ x (car li))) )
                            (else (even-odd-help (cdr li) (- x (car li))) )
) )
```

Problem 3:

```
(define (swap a b li) (swap-help a b li '() 0 (length li)) )

(define (indx-ret i li) (if (= i 0)
                            (list (car li))
                            (indx-ret (- i 1) (cdr li)) ))

(define (swap-help a b li li-temp indx-counter li-length)
  (cond ((= indx-counter li-length) li-temp)
        ((= a indx-counter) (swap-help a b li (append li-temp (indx-ret b
li)) (+ indx-counter 1) li-length))
        ((= b indx-counter) (swap-help a b li (append li-temp (indx-ret a
li)) (+ indx-counter 1) li-length) )
        (else (swap-help a b li (append li-temp (indx-ret indx-counter li)
) (+ indx-counter 1) li-length))) )
```