1.

2. Хвостовая рекурсия — частный случай рекурсии, при котором любой рекурсивный вызов является последней операцией перед возвратом из функции.
Вычисление факториала:
с хвостовой рекурсией:

```
(define (fact n)
(let loop ((i 1) (res 1))
 (if (>= i n) (* i res) (loop (add1 i) (* i res)))
        )
 )
```

без хвостовой рекурсии:

```
(define (fact n)
(if (<= n 1) 1
          (* n (fact (sub1 n))))
 )
```

3. Смысл мемоизации заключается в сохранении уже вычисленных результатов. При вызове той же функции сначала проверяется существование нужного значения в памяти, в случае, если его нет - значение вычисляется на основе уже имеющихся результатов, если оно есть - выдаем готовый результат, взятый из памяти. Мемоизация оправдана, если необходимо несколько раз вызвать одну и ту же функцию, причем для близких по значению аргументов. Рассмотрим пример с вычислением числа Фибоначчи. Если уже известно n-ое число Фибоначчи, то n+1 число считается за 1 шаг.

Пример: вычисление степеней тройки

```
(define pow-3-table (make-hash '((0 . 1))))
(define (pow-3 n)
 (if (hash-has-key? pow-3-table n) (hash-ref pow-3-table n)
    (let ((pow-n (* 3 (pow-3 (sub1 n)))))
      (begin (hash-set! pow-3-table n pow-n) pow-n)))
 )
```

Вызовы, когда есть выигрыш:

```
(define n 30)
(pow-3 n)
(pow-3 (add1 n))
```