Консультация по ФП (решения)

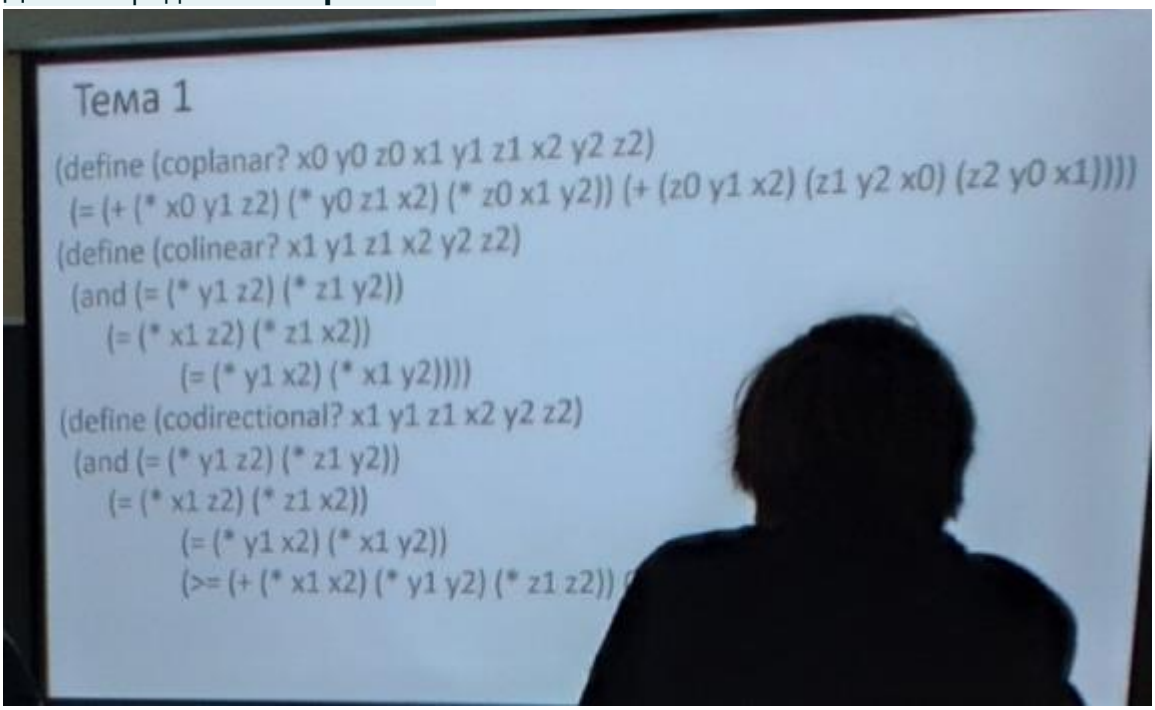
Формат файла:

пример задания (могут отличаться друг от друга)
решение с консы

- Анкета1

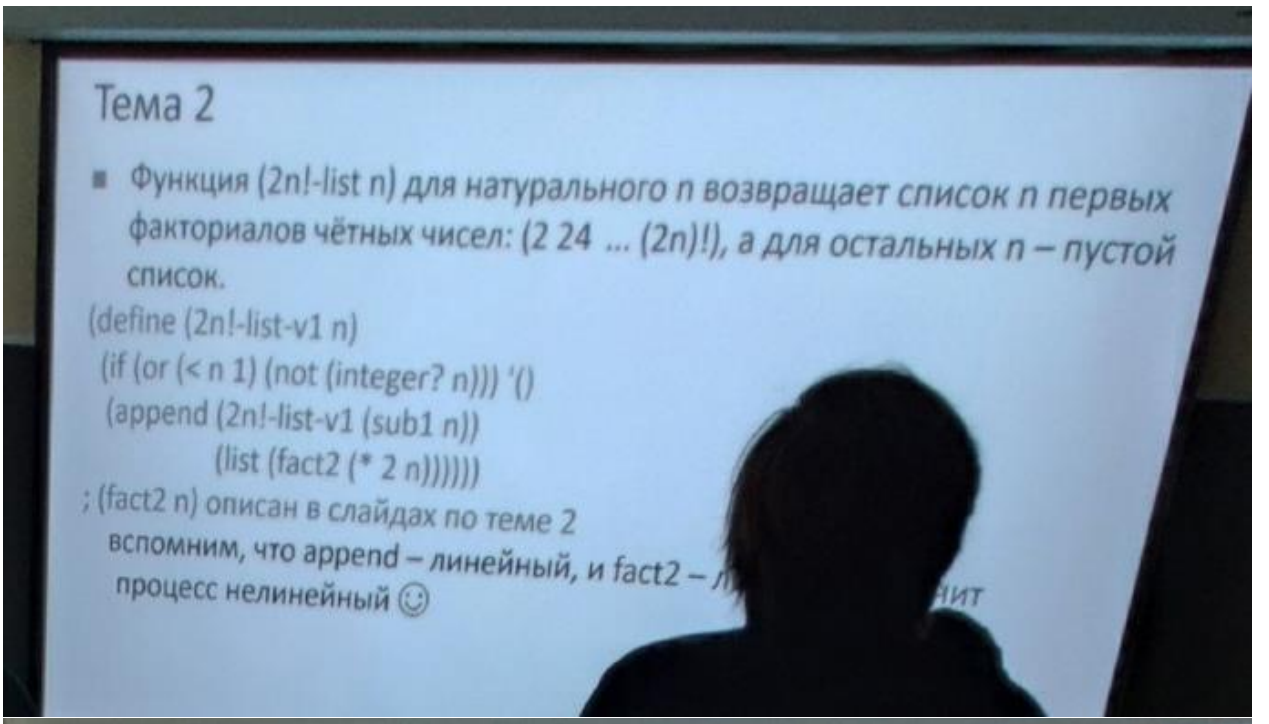
Функция **(coplanar? x0 y0 z0 x1 y1 z1 x2 y2 z2)** для целых чисел **x0**, **y0**, **z0**, **x1**, **y1**, **z1**, **x2**, **y2**, **z2** являющихся координатами трёх векторов **v0, v1** и **v2** в 3х-мерном пространстве, возвращает **#t**, если они компланарны, а иначе – **#f**.

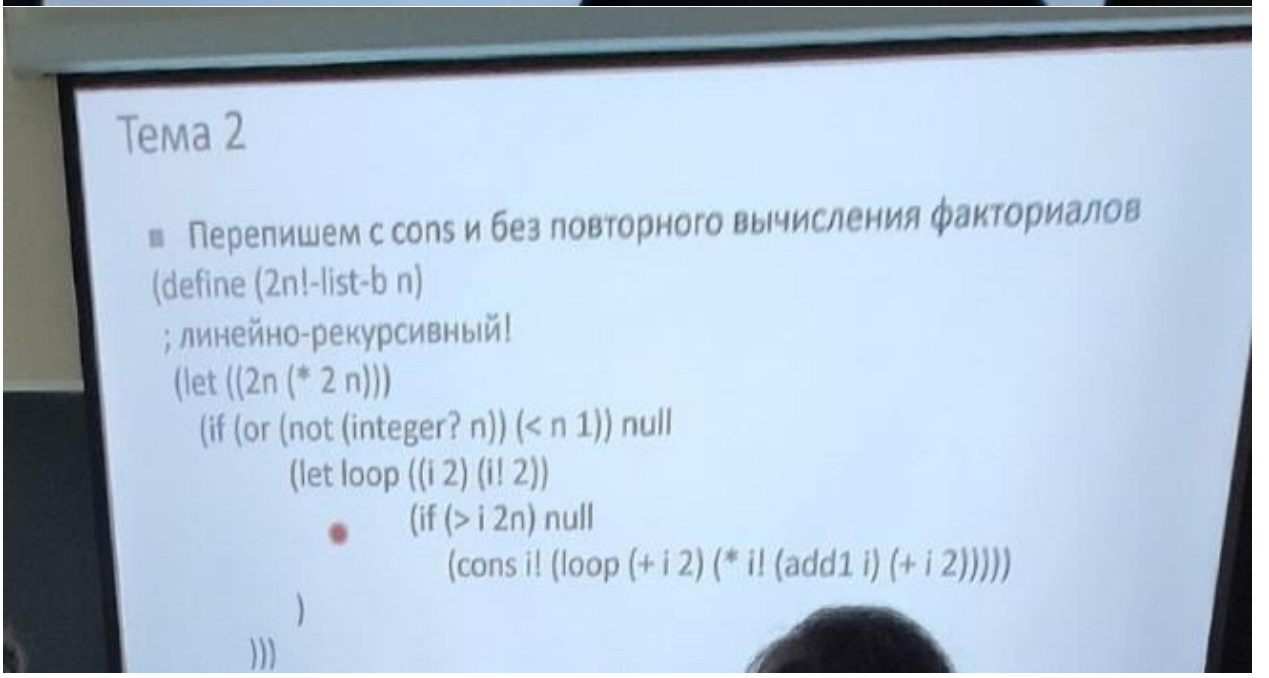
Дайте определение **coplanar?**:

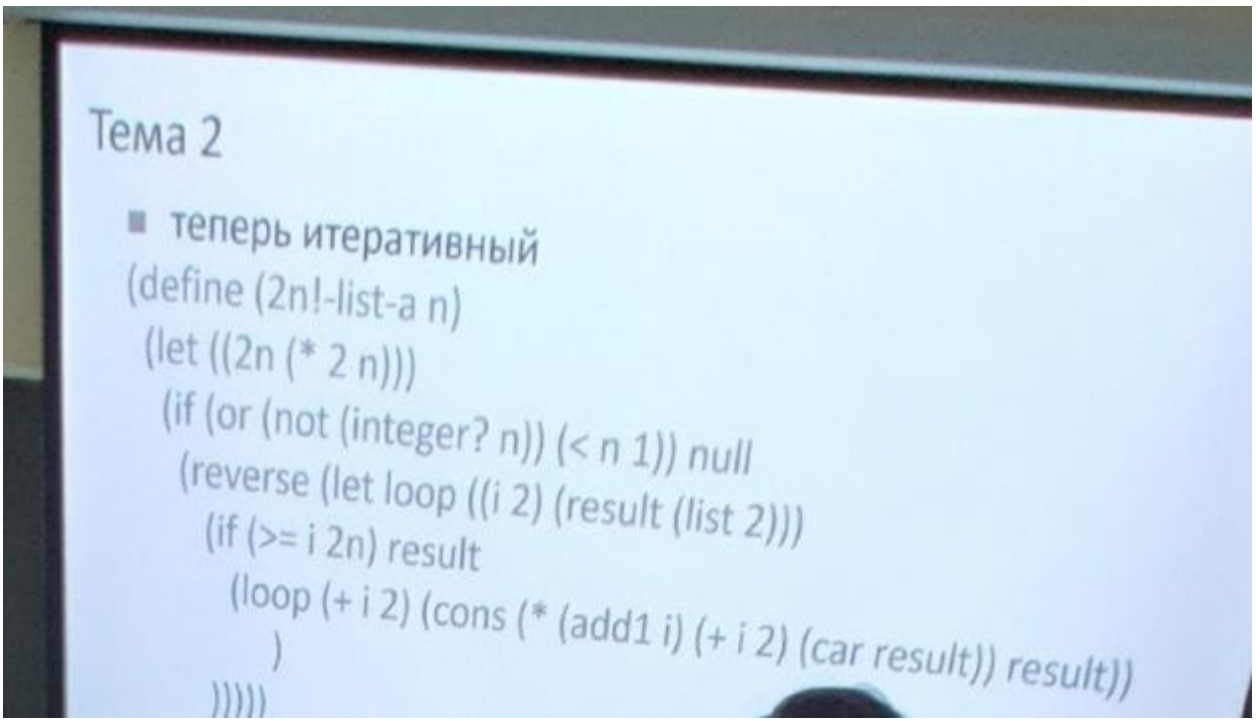


- Анкета2

Функция **(even-fib-list n)** для положительного натурального **n** возвращает список из первых **n** чисел в ряду *чётных* чисел Фибоначчи: **0 2 8 34 ...** , а для остальных **n** – пустой список. Числа в списке должны идти *по возрастанию*.
Дайте два определения этой функции:
a) определение, порождающее *линейно итеративный* процесс, записанное с именованным **let**;
b) определение, порождающее *линейно рекурсивный* процесс.


Тема 2
Функция (2n!-list n) для натурального n возвращает список n первых факториалов чётных чисел: (2 24 ... (2n)!), а для остальных n – пустой список.
(define (2n!-list-v1 n)
(if (or (< n 1) (not (integer? n))) '()
(append (2n!-list-v1 (sub1 n))
(list (fact2 (* 2 n))))))
; (fact2 n) описан в слайдах по теме 2
вспомним, что append – линейный, и fact2 –
процесс нелинейный ☺

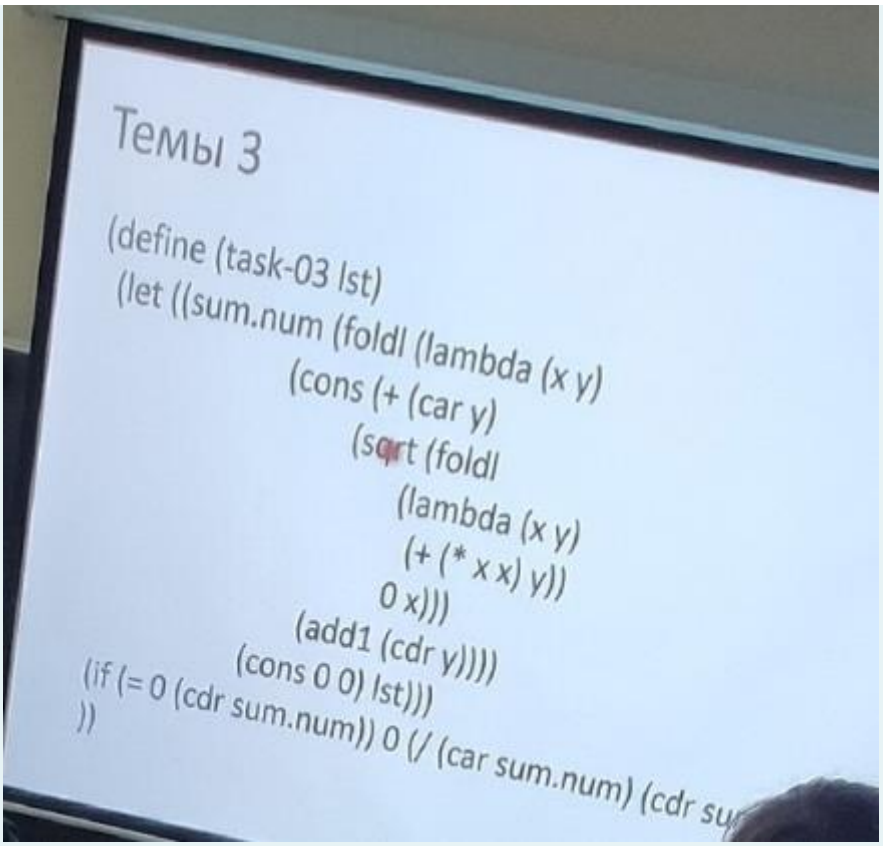

Тема 2
Перепишем с cons и без повторного вычисления факториалов
(define (2n!-list-b n)
; линейно-рекурсивный!
(let ((2n (* 2 n)))
(if (or (not (integer? n)) (< n 1)) null
(let loop ((i 2) (i! 2))
(if (> i 2n) null
(cons i! (loop (+ i 2) (* i! (add1 i) (+ i 2))))))
)
)))

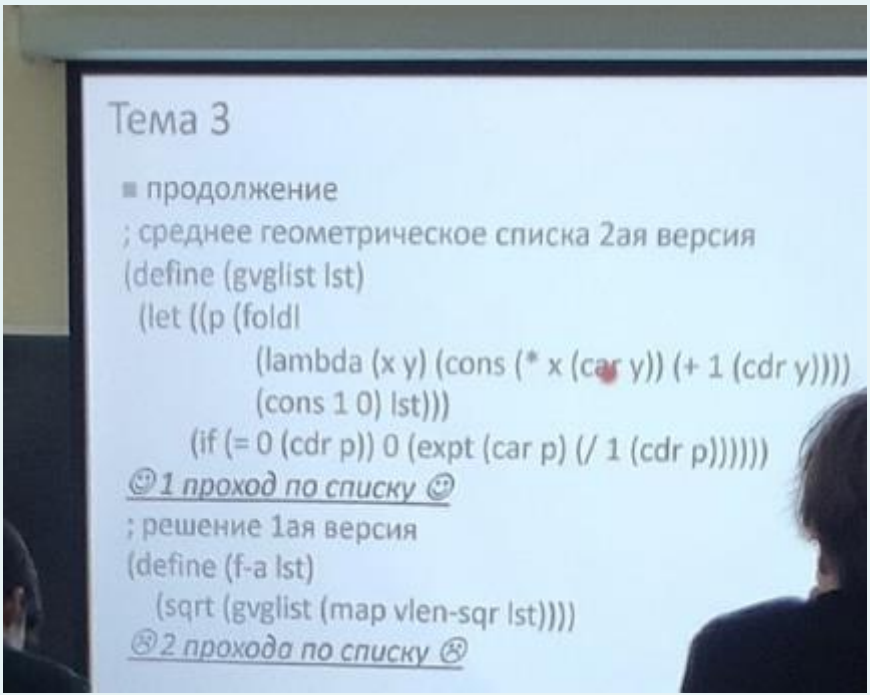


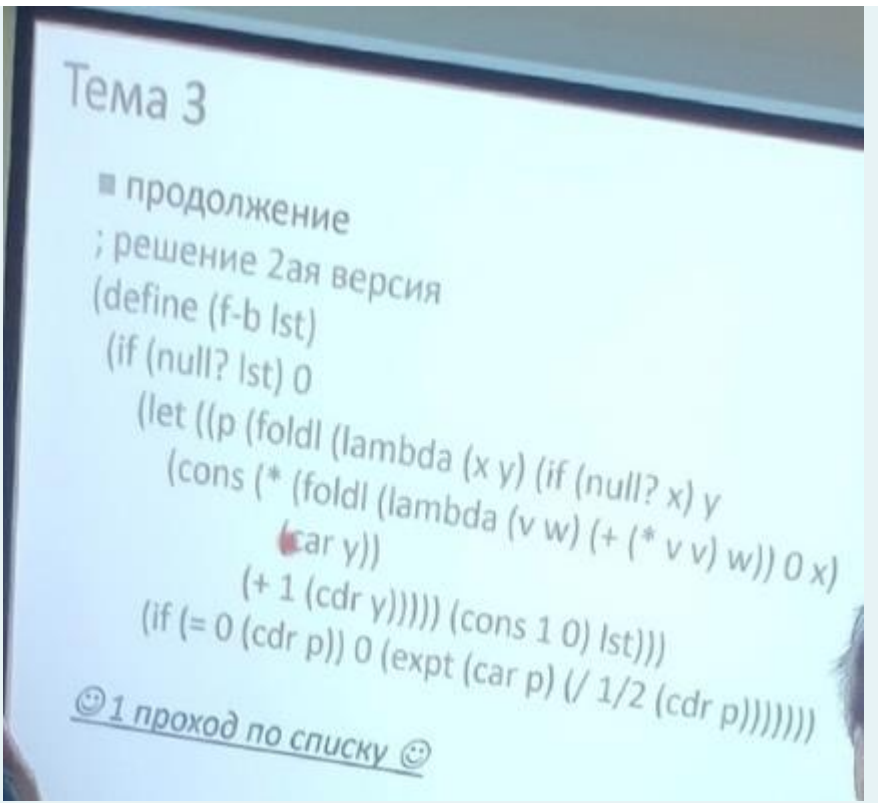
- Анкета3

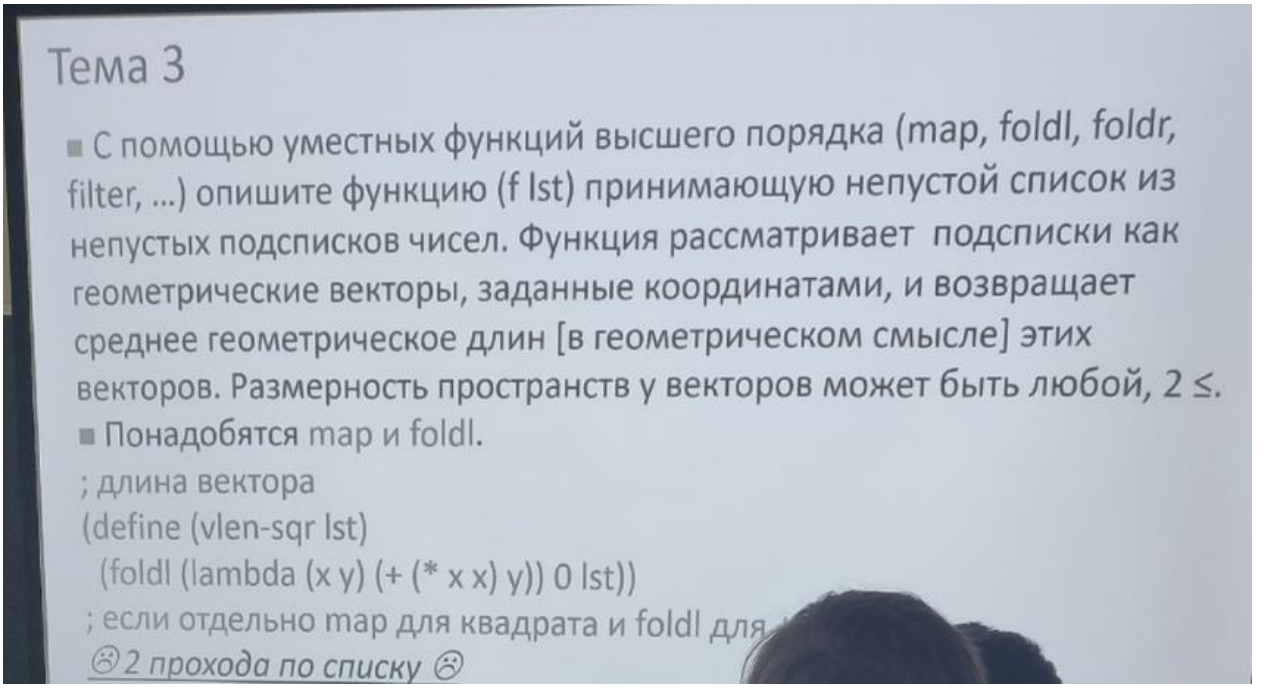
- С помощью уместных функций высшего порядка (**map**, **foldl**, **foldr**, **filter**, ...) опишите функцию (task-03 lst), принимающую непустой список из непустых подсписков неотрицательных чисел. Функция рассматривает подсписки как геометрические векторы, заданные координатами, и возвращает среднее арифметическое длин этих векторов [длины понимаются в геометрическом смысле, в алгебраическом смысле это нормы]. Размерность пространств у векторов натуральная, 2 ≤.
- 1) На слайде №22 (см. номер в правом нижнем углу слайда) дано определение функции **(list-fib-squares n)**, строящей список квадратов первых **n** чисел Фибоначчи (упорядоченных по возрастанию). Оно неэффективно.
  1.a) Дайте определение функции **(list-fib-squares-a n)** которое будет порождать линейно-итерационный процесс и будет использовать при этом уместные функции высшего порядка для обработки списков: **map**, **foldr**, **foldl**, ... (Вы сами должны выбрать подходящие).
  1.b) Дайте определение функции **(list-fib-squares-b n)** которое будет порождать линейно-итерационный процесс и будет использовать при этом только свёртку в качестве цикла, строящего требуемый список. Для построения вспомогательного сворачиваемого списка можете использовать функцию **(build-list n f)**, которая строит список из результатов применения функции **f** к числам 0, 1, ... n-1. См. доки Racket по этой функции.
- 2) Слайд №25. Функция **(process lst)** получает непустой список списков чисел **lst** и возвращает список, составленный из следующих по порядку тех элементов **lst**, у которых сумма их элементов больше произведения элементов списка, являющегося начальным элементом **lst**. Пусть сумма пустого списка равна 0. Пусть произведение пустого списка равно 1. Реализация должна быть

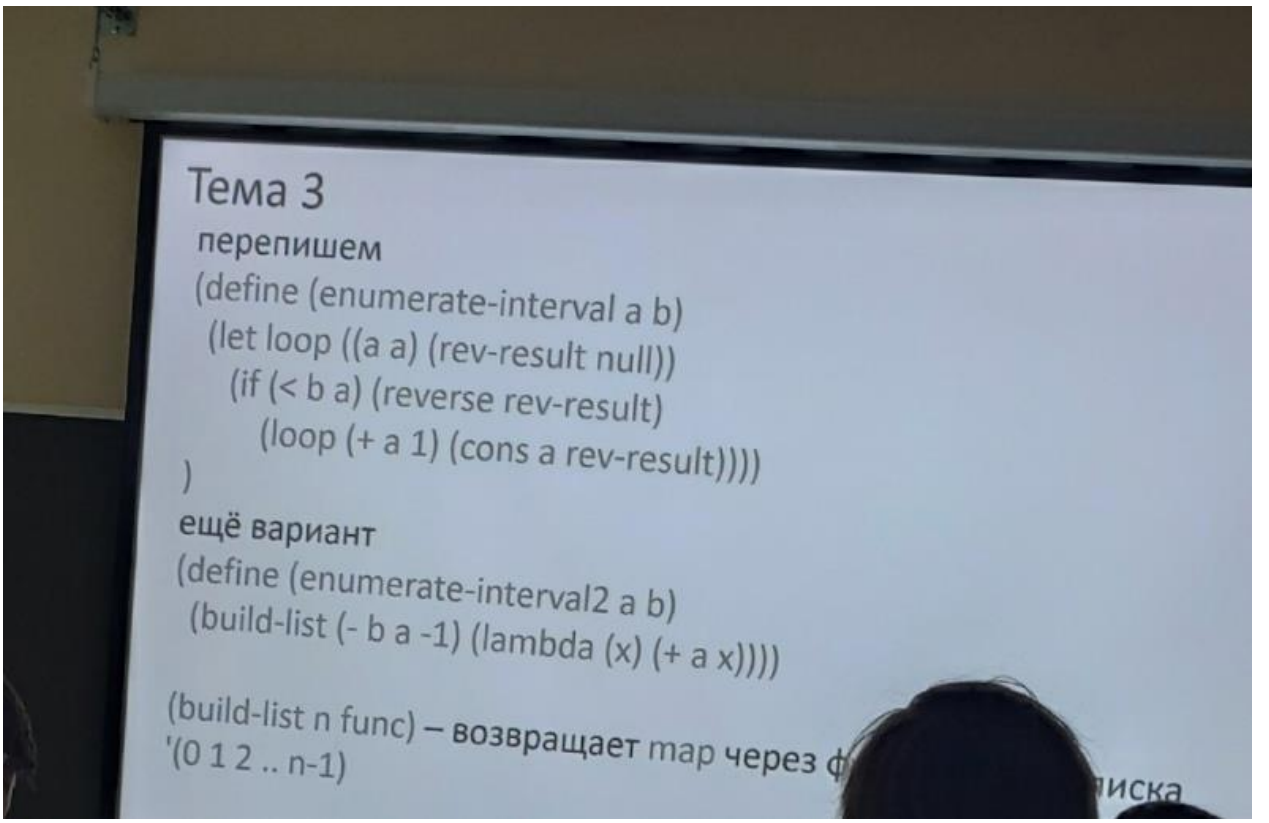
эффективной и использующей уместные функции высшего порядка для обработки списков.

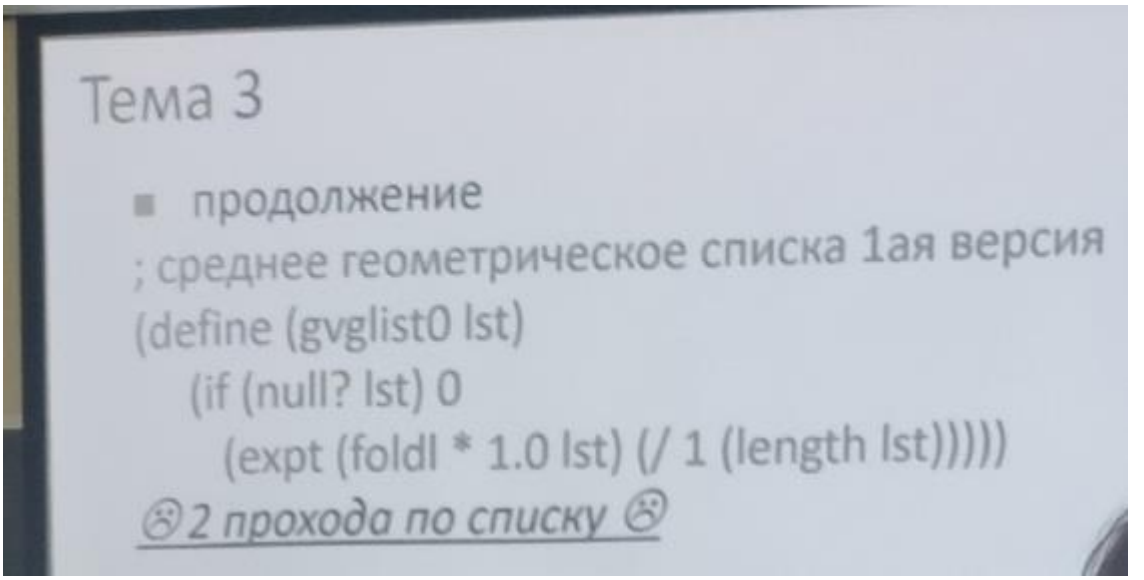





Тема 3
продолжение
; решение 2ая версия
(define (f-b lst)
(if (null? lst) 0
(let ((p (foldl (lambda (x y) (if (null? x) y
(cons (* (foldl (lambda (v w) (+ (* v v) w)) 0 x)
(car y))
(+ 1 (cdr y))))))) (cons 1 0) lst)))
(if (= 0 (cdr p)) 0 (expt (car p) (/ 1/2 (cdr p)))))))
☺1 проход по списку ☺


Тема 3
С помощью уместных функций высшего порядка (map, foldl, foldr, filter, ...) опишите функцию (f lst) принимающую непустой список из непустых подсписков чисел. Функция рассматривает подсписки как геометрические векторы, заданные координатами, и возвращает среднее геометрическое длин [в геометрическом смысле] этих векторов. Размерность пространств у векторов может быть любой, 2 ≤.
Понадобятся map и foldl.
; длина вектора
(define (vlen-sqr lst)
(foldl (lambda (x y) (+ (* x x) y)) 0 lst))
; если отдельно map для квадрата и foldl для
☹2 прохода по списку ☹


Тема 3
перепишем
(define (enumerate-interval a b)
(let loop ((a a) (rev-result null))
(if (< b a) (reverse rev-result)
(loop (+ a 1) (cons a rev-result))))
)
ещё вариант
(define (enumerate-interval2 a b)
(build-list (- b a -1) (lambda (x) (+ a x))))
(build-list n func) – возвращает map через ф
'(0 1 2 .. n-1)
иска


Тема 3
продолжение
; среднее геометрическое списка 1ая версия
(define (gvglist0 lst)
(if (null? lst) 0
(expt (foldl * 1.0 lst) (/ 1 (length lst)))))
☹2 прохода по списку ☹

Тема 3

Эффективный список квадратов первых n чисел Фибоначчи *без высших функций*:

```
(define (list-fib-squares n)
  (let loop ((i n) (fn-1 1) (fn-2 0) (rev-result null))
    (if (<= i 0) (reverse rev-result)
        (loop (- i 1) (+ fn-1 fn-2) fn-1 (cons (* fn-2 fn-2) rev-result))))
)
```

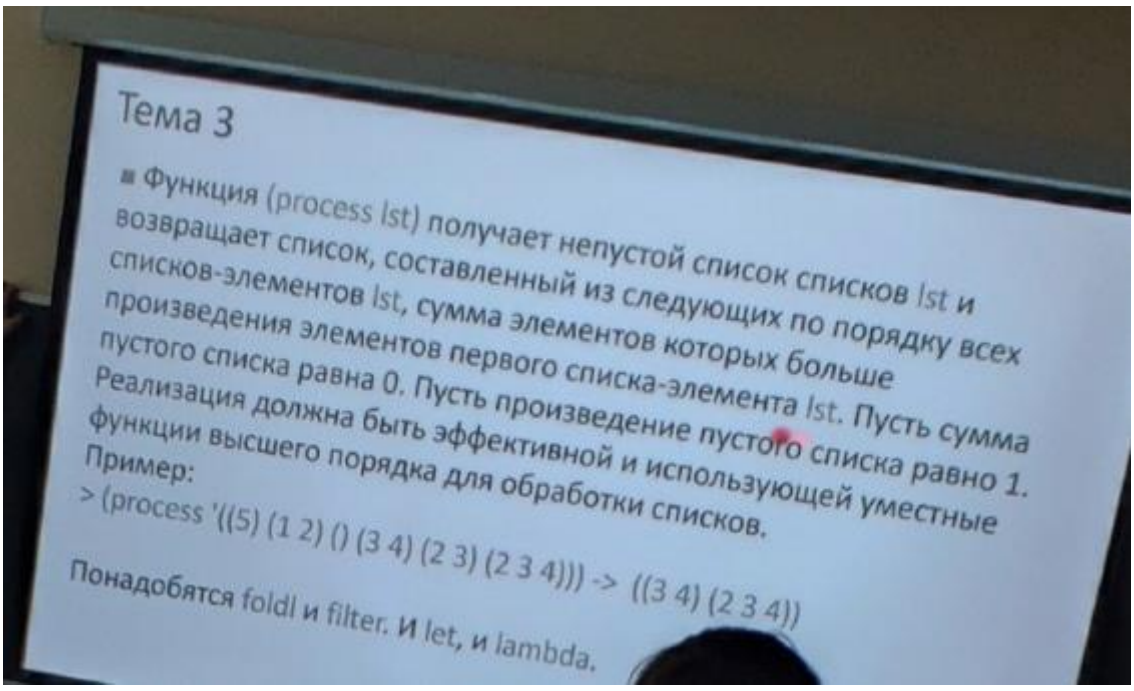
Тема 3

с высшей функцией build-list

```
(define (list-fib-squares1 n)
    (define (fib n fn-1 fn-2)
      (if (<= n 0) fn-1 (fib (- n 1) (+ fn-1 fn-2) fn-1)))
  (build-list n (lambda (x) (let ((temp  (fib x 1 0))) (* temp temp)))))
```

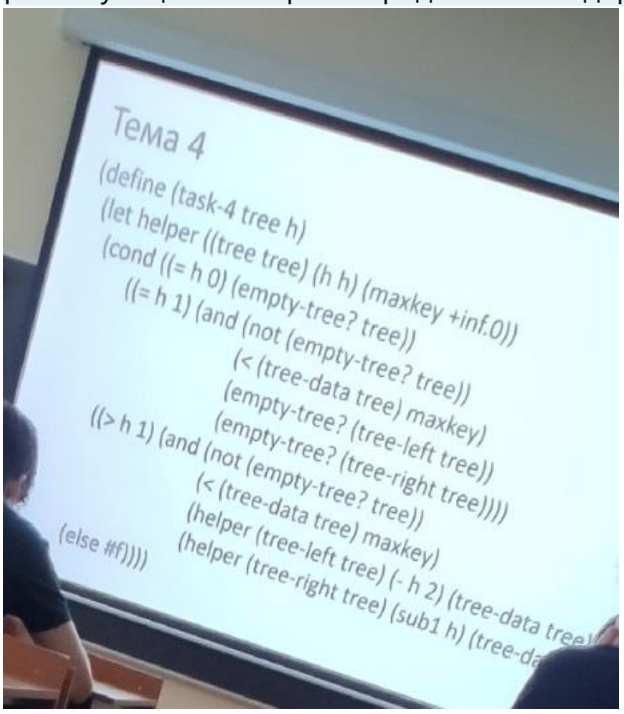
со свёрткой ☺

```
(define (list-fib-squares2 n)
  (reverse (cddr (foldl     ; приём – в результате свёртки копим
                  (lambda (x y)  ; помимо ответа «рабочие данные»
                    (let* ((fn-2 (cadr y)) (fn-1 (car y)) (fn (+ fn-1 fn-2)))
                      (list* fn fn-1 (* fn-1 fn-1) (cddr y))))
                  '(1 0) (build-list n (lambda (x) 1))))))
```



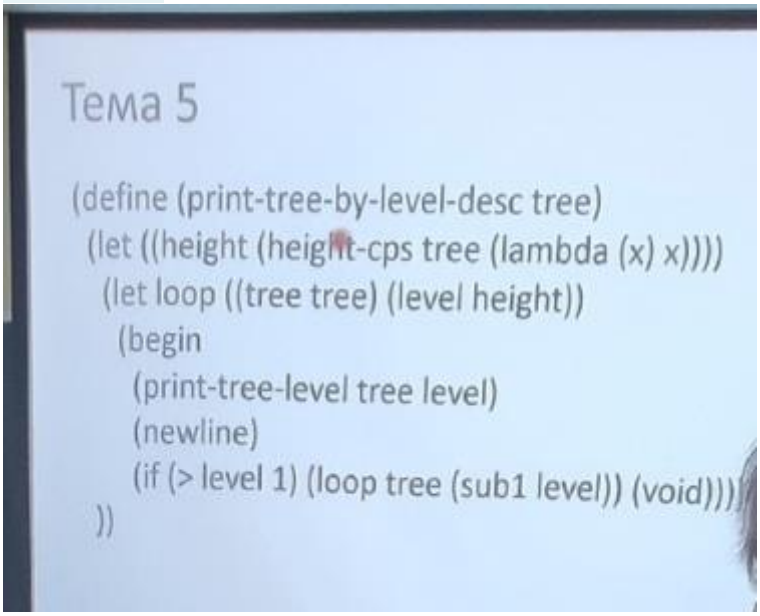
- Анкета4

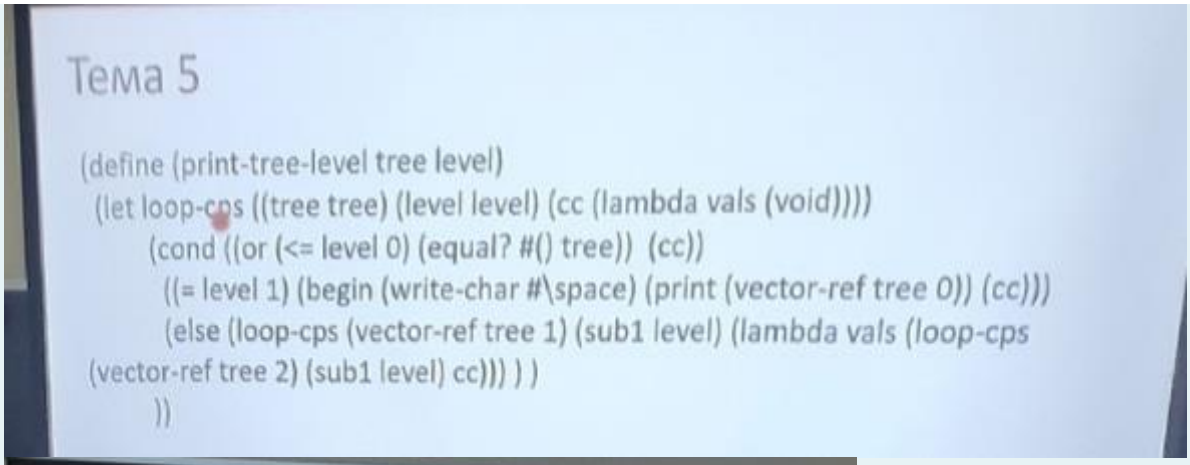
  Опишите функцию **(task-4 tree h)**, проверяет, является ли двоичное дерево tree деревом-кучей высоты h. Положим, что пустое дерево является деревом-кучей высоты 0, дерево из одного листа -- дерево-куча высоты 1, дерево из корня, у которого левое поддерево является деревом-кучей высоты hl, а правое -- деревом-кучей высоты hr и |hl - hr|<2 -- дерево-куча высоты max(hl, hr) + 1. (разность высот потомков по модулю небольше 1). Для каждой вершины дерева-кучи справедливо, что данные при вершине являются числом и это число больше, чем любое число в поддеревьях этой вершины.
  В решении используйте функции работы с деревьями, введённые на лекции, реализующие векторное представление дерева.

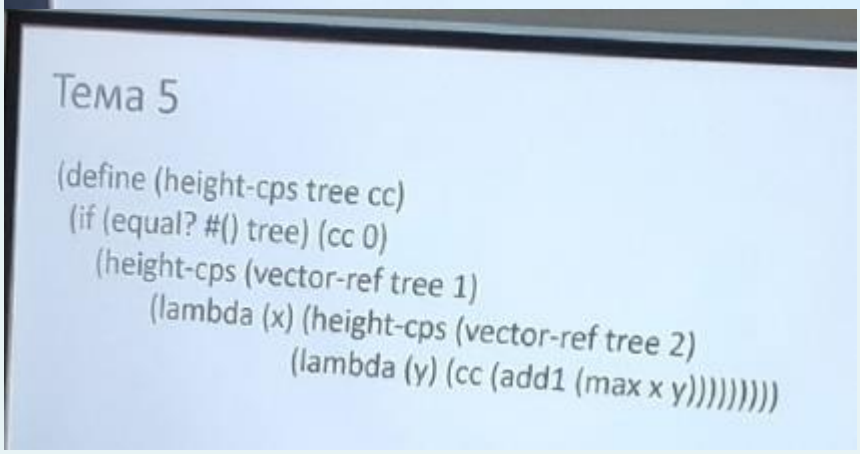


- Анкета5

  Опишите функцию **(print-tree-by-level-desc tree)**. Функция получает двоичное дерево в векторном представлении и печатает его по уровням -- каждый уровень на отдельной строке, значения при вершинах уровня *справа налево* через пробелы -- в порядке убывания номеров уровней. Используйте стиль передачи остаточных вычислений, чтобы всюду рекурсия была хвостовой.

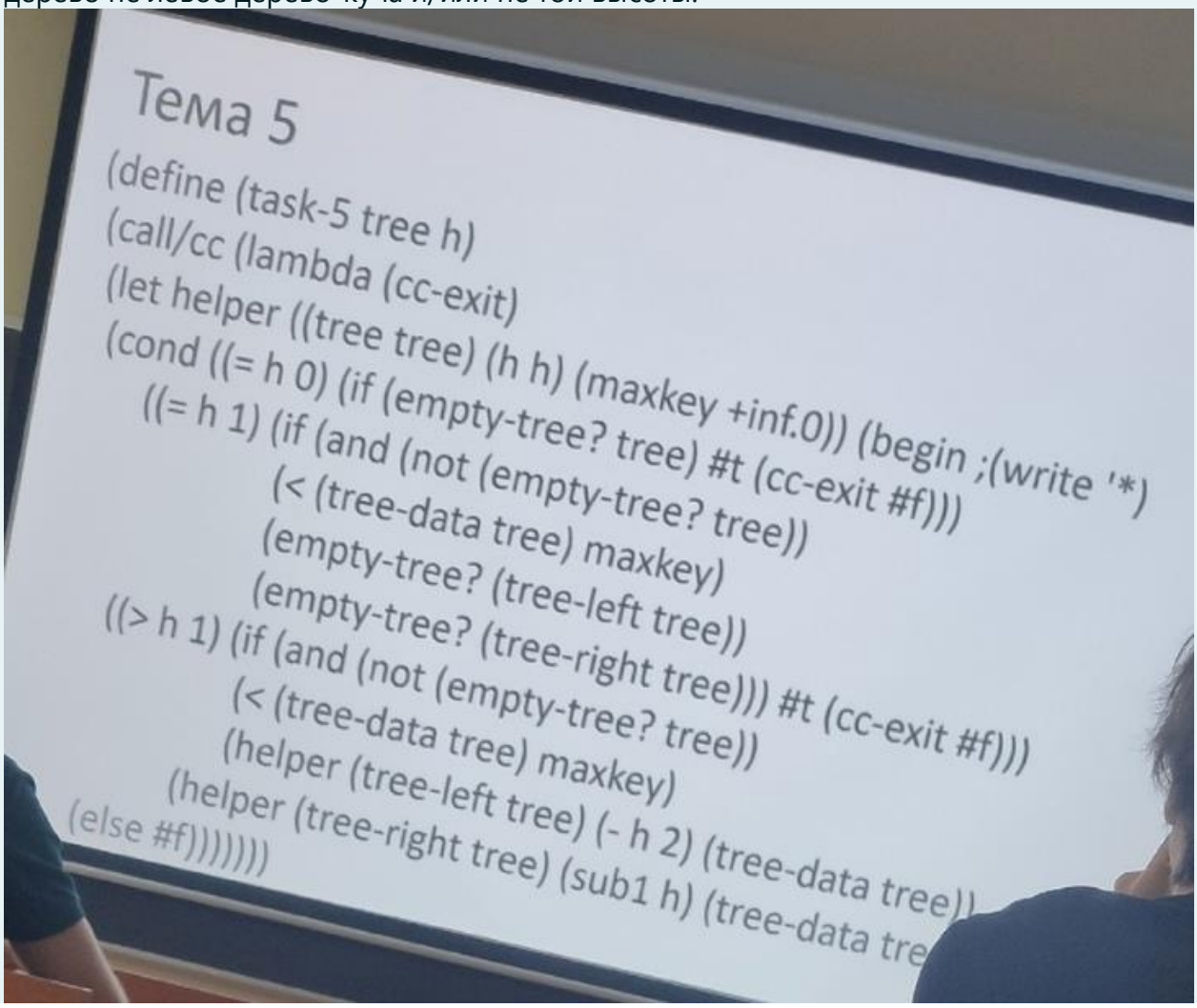






Опишите функцию **(task-5 tree h)**, проверяет, является ли двоичное дерево tree левым деревом-кучей высоты h. Положим, что пустое дерево является левым деревом-кучей высоты 0, дерево из одного листа -- левое дерево-куча высоты 1, дерево из корня, у которого левое поддерево является левым деревом-кучей высоты 1, а правое -- левым деревом-кучей высоты 0 -- левое дерево-куча высоты 2 и т. д. (левый потомок ровно на 1 выше правого и оба -- левые деревья-кучи). Для каждой вершины дерева-кучи справедливо, что данные при вершине являются числом и это число больше, чем любое число в поддеревьях этой вершины.

В решении используйте функции работы с деревьями, введённые на лекции, реализующие векторное представление дерева. С помощью **call/cc** добейтесь эффективного решения, чтобы #f возвращалось сразу, как только становится ясно, что дерево не левое дерево-куча и/или не той высоты.



Демо тест

на лекции Малышко не показывал ответы (ну либо мы их не сфоткали, так что верных ответов на 5,8,9 мы не имеем)

Вопрос **1**
Выполнен
Баллов: 0,50 из 0,50
Отметить вопрос

Какое подвыражение вычисляется первым при вычислении выражения **(f a b)** при выполнении *согласно стандарту языка*?

- [x] a. неизвестно, так как по стандарту порядок вычисления подвыражений не фиксирован и определяется в реализации интерпретатора
- [ ] b. **a**
- [ ] c. **f**
- [ ] d. **b**

Вопрос **2**
Выполнен
Баллов: 0,50 из 0,50
Отметить вопрос

Укажите среди предложенного те и только те выражения, которые *являются литералами*.

- [x] a. **2/3**
- [ ] b. **(/ 2 3)**
- [x] c. **(quote (/ 2 3))**
- [x] d. **'(/ 2 3)**
- [ ] e. **+**
- [ ] f. **('a 'b 'c)**
- [ ] g. **abc**
- [ ] h. **(+ 1 (+ 2 3))**
- [x] i. **'abc**
- [x] j. **"abc"**
- [ ] k. **(a b c)**
- [ ] l. **(a 'b 'c)**
- [x] m. **#t**

Вопрос **3**
Выполнен
Баллов: 0,50 из 0,50
Отметить вопрос

Укажите, что не так в определении **(define cube (x) (* x x x))**?

- [x] a. следует вместо **cube (x)** писать **(cube x)**
- [ ] b. имя **cube** переопределять нельзя
- [ ] c. у функции * может быть только два параметра, а здесь указаны три
- [ ] d. всё верно, ошибок нет
- [ ] e. отсутствует оператор возврата
- [ ] f. вместо **define** нужно писать **def**
- [ ] g. такая функция может быть описана *только* как анонимная, т. е. через спецформу **lambda**

Вопрос **4**
Выполнен
Баллов: 0,50 из 0,50
Отметить вопрос

Положим, что пользователь определил функцию **(define (quadruplicate x) (+ x x (+ x x)))**, которая находит учетверённое значение своего параметра **x** суммированием. Укажите, сколько вызовов функции + потребуется выполнить при нормальном порядке вычислений (точнее, том порядке, который назван "нормальным" в учебнике), чтобы узнать значение выражения **(quadruplicate (quadruplicate 2))**?

- [ ] a. 4
- [ ] b. 6
- [ ] c. 9
- [ ] d. 7
- [ ] e. 3
- [ ] f. 2
- [x] g. 10
- [ ] h. 5
- [x] i. количество суммирований будет чётным
- [ ] j. больше 10
- [ ] k. 8
- [ ] l. количество суммирований будет нечётным

Вопрос **5**
Выполнен
Баллов: 0,00 из 0,50
Отметить вопрос

Укажите среди перечисленного то и только то, что справедливо для специальной формы **cond**.

- [x] a. в **cond** предикат из очередной альтернативы вычисляется только тогда, когда либо он первый, либо все предшествующие ему были ложны
- [ ] b. в **cond** всегда должна быть **else** альтернатива
- [x] c. в **cond** предикат из очередной альтернативы не вычисляется только тогда, когда хотя бы один из предшествующих ему был равен **#t**
- [x] d. в **cond** всегда должна быть либо одна альтернатива, либо более чем одна альтернатива
- [ ] e. если в **cond** истинны предикаты в нескольких альтернативах, то выполняемая альтернатива выбирается из них случайно
- [ ] f. в **cond** всегда должны быть либо две альтернативы, либо более чем две альтернативы

Вопрос **6**
Выполнен
Баллов: 0,50 из 0,50
Отметить вопрос

Укажите среди перечисленного то и только то, что справедливо для рекурсивных функций.

- [ ] a. вызов рекурсивной функции не может породить рекурсивный процесс
- [ ] b. отложенные вычисления возникают при вычислении вызовов рекурсивных функций только в случае хвостовой рекурсии
- [ ] c. вызовы рекурсивных функций порождают только рекурсивные процессы
- [x] d. вызов рекурсивной функции может породить рекурсивный процесс
- [x] e. вызов рекурсивной функции может породить итеративный процесс
- [ ] f. отложенные вычисления возникают всегда при вычислении вызовов рекурсивных функций
- [ ] g. вызов рекурсивной функции не может породить итеративный процесс

Вопрос **7**
Выполнен
Баллов: 0,50 из 0,50
Отметить вопрос

Реализованное на языке Scheme эффективное итеративное вычисление списка, составленного из факториалов первых **N** натуральных чисел, идущих по убыванию, требует... (укажите лучшую оценку из подходящих)

- [ ] a. константной памяти, не зависящей от **N**
- [ ] b. квадратичной относительно **N** памяти
- [ ] c. экспоненциального относительно **N** времени счёта
- [ ] d. логарифмического относительно **N** времени счёта
- [ ] e. экспоненциальной относительно **N** памяти
- [ ] f. квадратичного относительно **N** времени счёта
- [x] g. линейной относительно **N** памяти
- [ ] h. константного времени счёта, не зависящего от **N**
- [x] i. линейного относительно **N** времени счёта

Вопрос **8**
Выполнен
Баллов: 0,00 из 0,50
Отметить вопрос

Дано определение **(define f (lambda (x) (f (+ 1 x))))**. Какой процесс будет порождён при вычислении вызова **(f 100)**?

- [x] a. рекурсивный
- [ ] b. итеративный
- [ ] c. никакой, так как в определении синтаксическая ошибка
- [x] d. зацикливающийся

Вопрос **9**
Выполнен
Баллов: 0,00 из 0,50
Отметить вопрос

Что сделать, чтобы программным способом убедиться в том, что **eval** в Scheme является функцией?

- [ ] a. переопределить **eval**
- [ ] b. вычислить вызов некоторой функции, указав **eval** среди аргументов вызова
- [ ] c. вызвать **eval** внутри вызова **eval**
- [ ] d. **eval** не является функцией, это спецформа
- [ ] e. в этом нельзя убедиться программным способом
- [ ] f. вычислить вызов **eval**, указав среди аргументов вызов некоторой функции
- [x] g. использовать **eval** внутри тела определения некоторой функции

Вопрос **10**
Выполнен
Баллов: 0,50 из 0,50
Отметить вопрос

Укажите значение выражения **((let ((t /)) (lambda (n) (lambda (x) (t x n)))) 2)**

- [x] a. анонимная функция, которая делит свой параметр на 2
- [ ] b. список из трёх элементов **(t 2 n)**
- [ ] c. ничего, так как в выражении ошибка
- [ ] d. анонимная функция, которая делит свой второй параметр на свой первый параметр
- [ ] e. список из трёх элементов **(t x 2)**
- [ ] f. анонимная функция, которая делит свой первый параметр на свой второй параметр
- [ ] g. анонимная функция, которая делит 2 на свой параметр

Upd: фулл9 и половина8

Вопрос 8
Выполнен
Баллов: 0,25 из 0,50
Отметить вопрос

Дано определение **(define f (lambda (x) (f (+ 1 x))))**. Какой процесс будет порождён при вычислении вызова **(f 100)**?

- a. никакой, так как в определении синтаксическая ошибка
- b. рекурсивный
- c. зацикливающийся
- d. итеративный

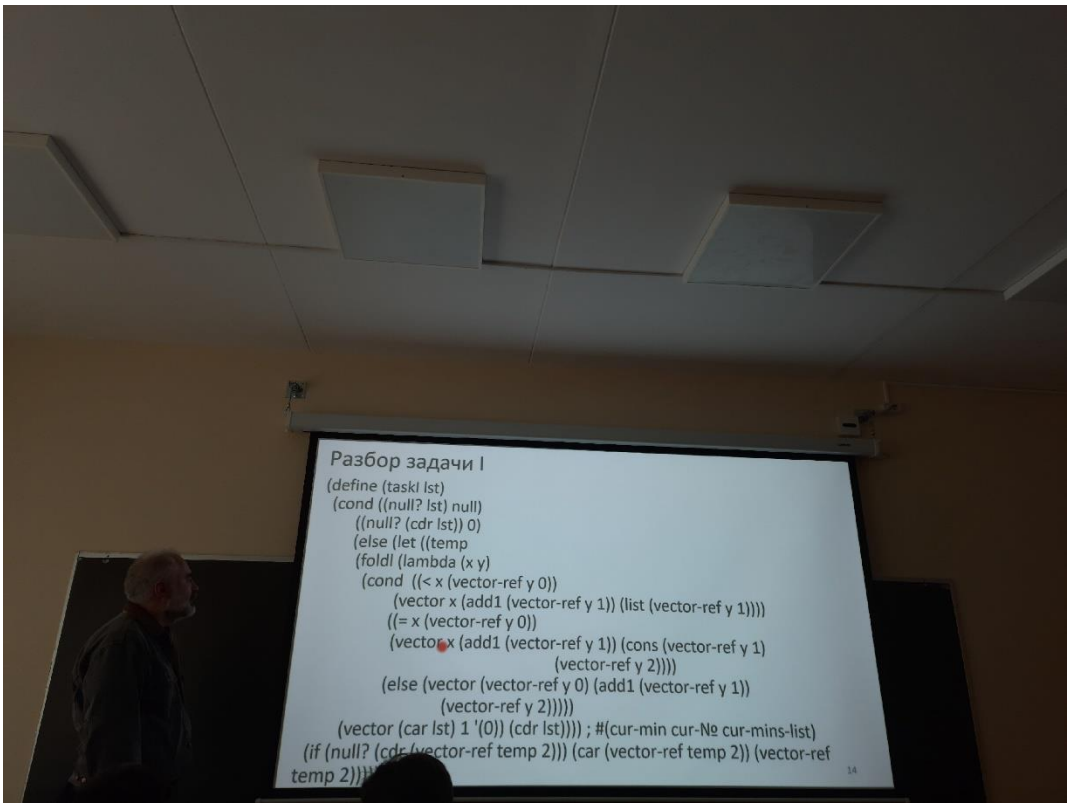
Вопрос 9
Выполнен
Баллов: 0,50 из 0,50
Отметить вопрос

Что сделать, чтобы программным способом убедиться в том, что **eval** в Scheme является функцией?

- a. **eval** не является функцией, это спецформа
- b. в этом нельзя убедиться программным способом
- c. использовать **eval** внутри тела определения некоторой функции
- d. переопределить **eval**
- e. вычислить вызов **eval**, указав среди аргументов вызов некоторой функции
- f. вычислить вызов некоторой функции, указав **eval** среди аргументов вызова
- g. вызвать **eval** внутри вызова **eval**

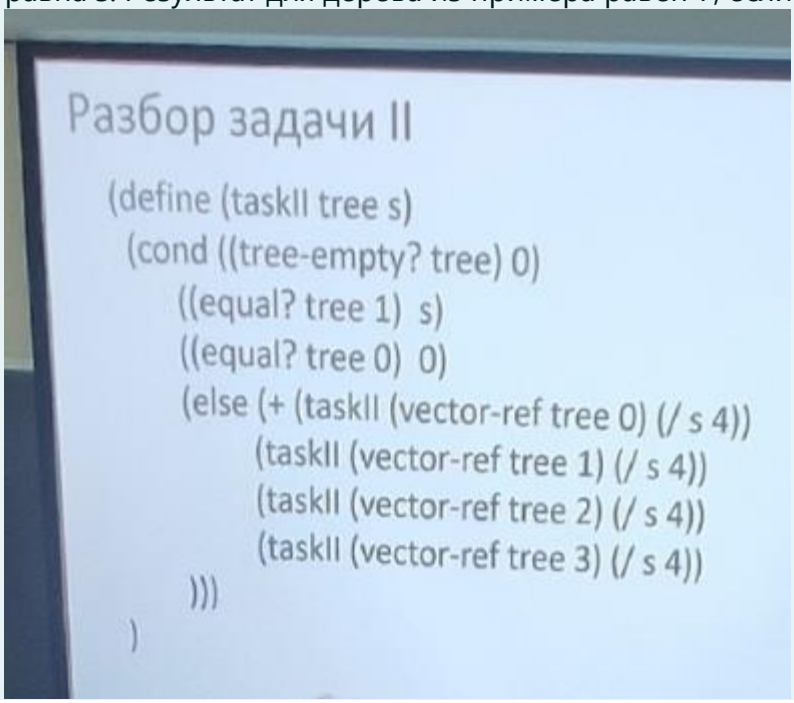
Демо11-1

I. Реализуйте функцию **(taskI lst)**, которая принимает список чисел и возвращает список номеров минимального элемента списка (если минимум в списке один, то просто список состоит из одного номера). Нумерация элементов списка начинается с нуля. Элементы списка-результата идут по убыванию. Если исходный список пуст, выдаётся пустой список. Выберите и обязательно используйте в своём решении уместные функции высших порядков для работы со списками, такие как **filter**, **map**, **foldl** и т. п. Пример: **(taskI (list -1 0 1 -1 0 1 -1)** => **(6 3 0)**
*Указание к задаче I: Каково минимальное количество проходов по **lst** для решения?*



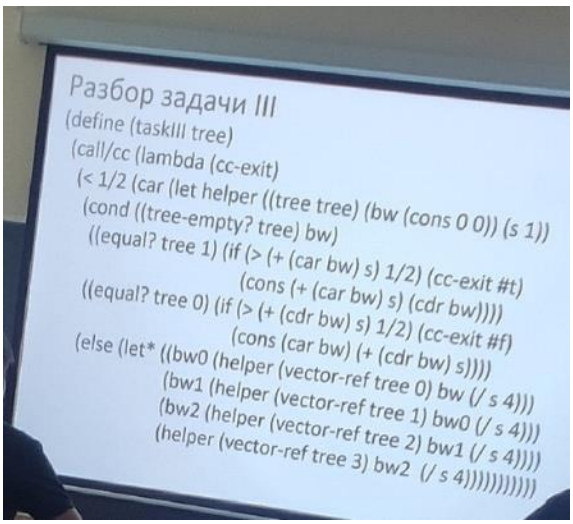
Демо11-2

II. Квадродеревом называется следующий способ представления растровых черно-белых изображений:
1) Если изображение целиком белое, то оно представляется квадродеревом из одной «белой» вершины. Значение такого квадродерева: 0 [ноль].
2) Если изображение целиком чёрное, то оно представляется квадродеревом из одной «чёрной» вершины. Значение такого квадродерева: 1[единица].
3) Если на изображении есть и чёрные и белые участки, то оно делится на 4 равные части (верхнюю левую, верхнюю правую, нижнюю левую, нижнюю правую) и представляется квадродеревом, состоящим из корневой вершины и четырёх поддеревьев, которые описывают части изображения. Обозначим значения поддеревьев: <верхлевд>, <верхправд>, <нижнлевд>, <нижнправд>; тогда значением всего дерева будет вектор из четырёх элементов, т. е. результат вычисления **(vector** <верхлевд> <верхппправд> <нижнлевд> <нижнправд>**)**.
Пример квадродерева: #(1 0 0 #(1 1 1 0))
Составьте функцию **(taskII t s)**, которая находит суммарную площадь чёрных участков изображения, считая, что площадь всего участка, описываемого квадродеревом **t**, равна **s**. Результат для дерева из примера равен 7, если площадь квадрата равна 16.



Демо11-3

III. Реализуйте функцию **(taskIII t)**, которая возвращает **#t**, если суммарная площадь черных участков квадродерева **t** больше, чем суммарная площадь его белых участков. Иначе **(taskIII t)** возвращает **#f**. Обязательно предусмотрите эффективную работу функции в тех случаях, когда рассмотренной части дерева уже достаточно для ответа и просмотр дальнейшей части не нужен.
Пример: **(taskIII #(1 1 0 #(1 1 1 0)))** => **#t (taskIII #(0 0 0 #(1 0 1 1))) => #f**



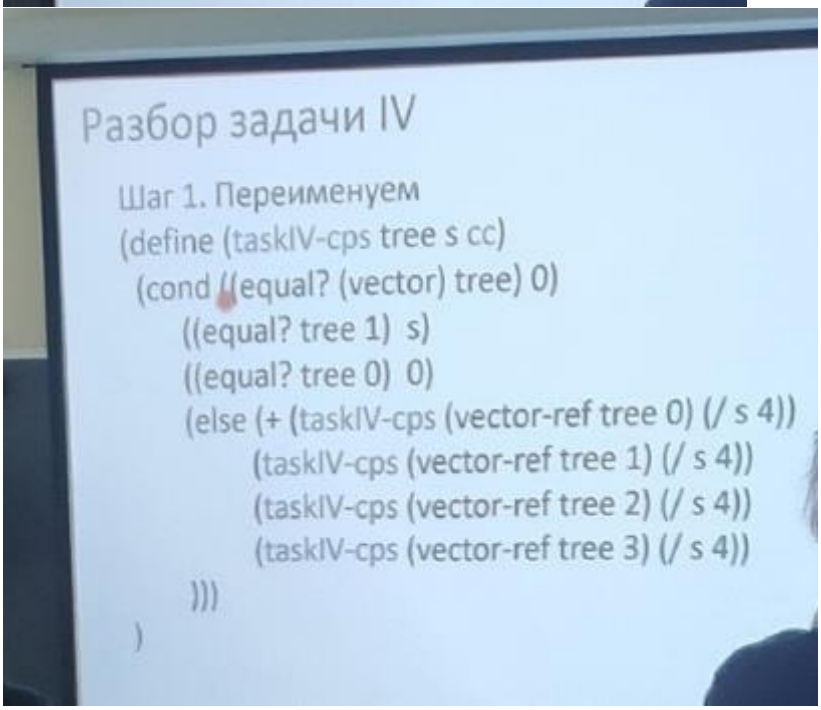
Демо11-4

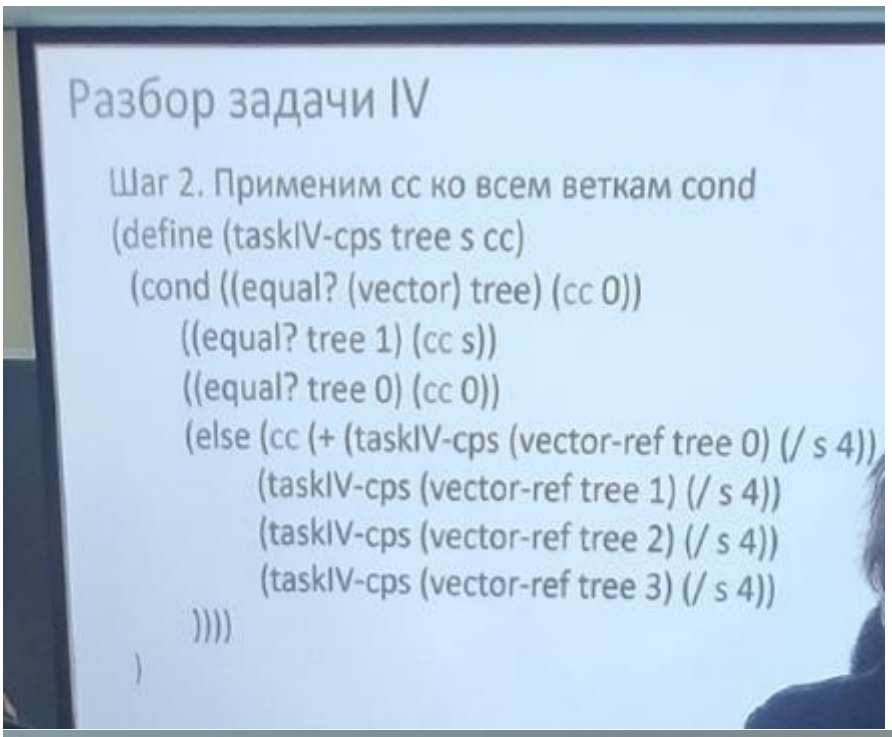
IV. Реализуйте функцию **(taskIV-cc t s cc)**, являющуюся версией **(taskII t s)**, которая составлена в стиле передачи продолжений.

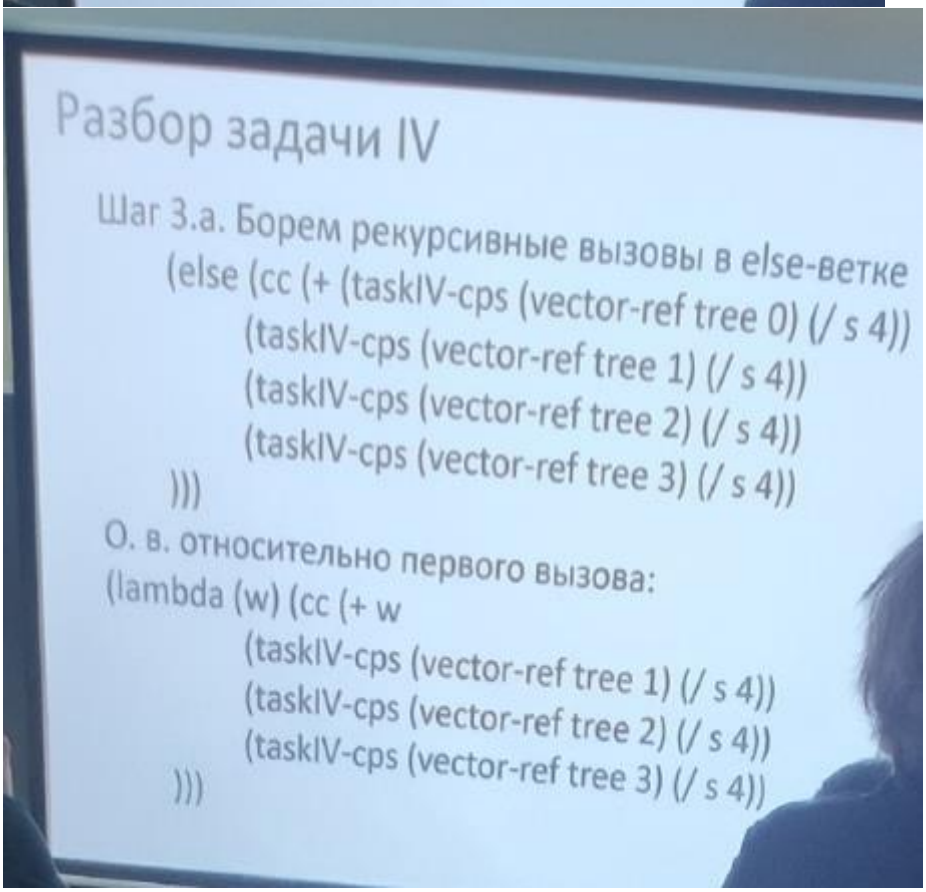
Разбор задачи IV

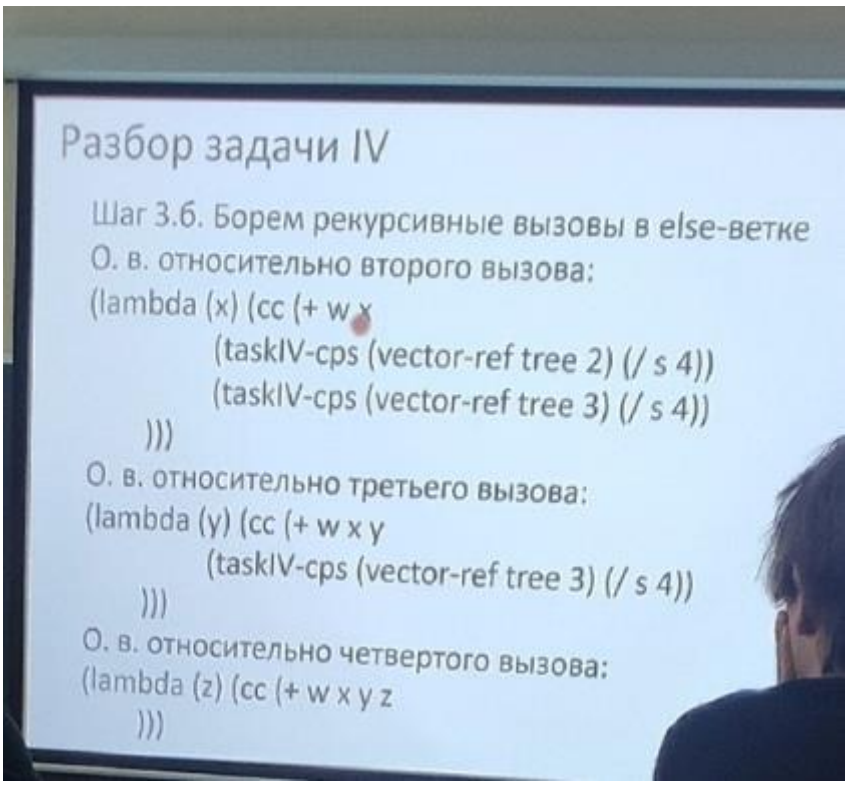
Берём код решения задачи II и правим его

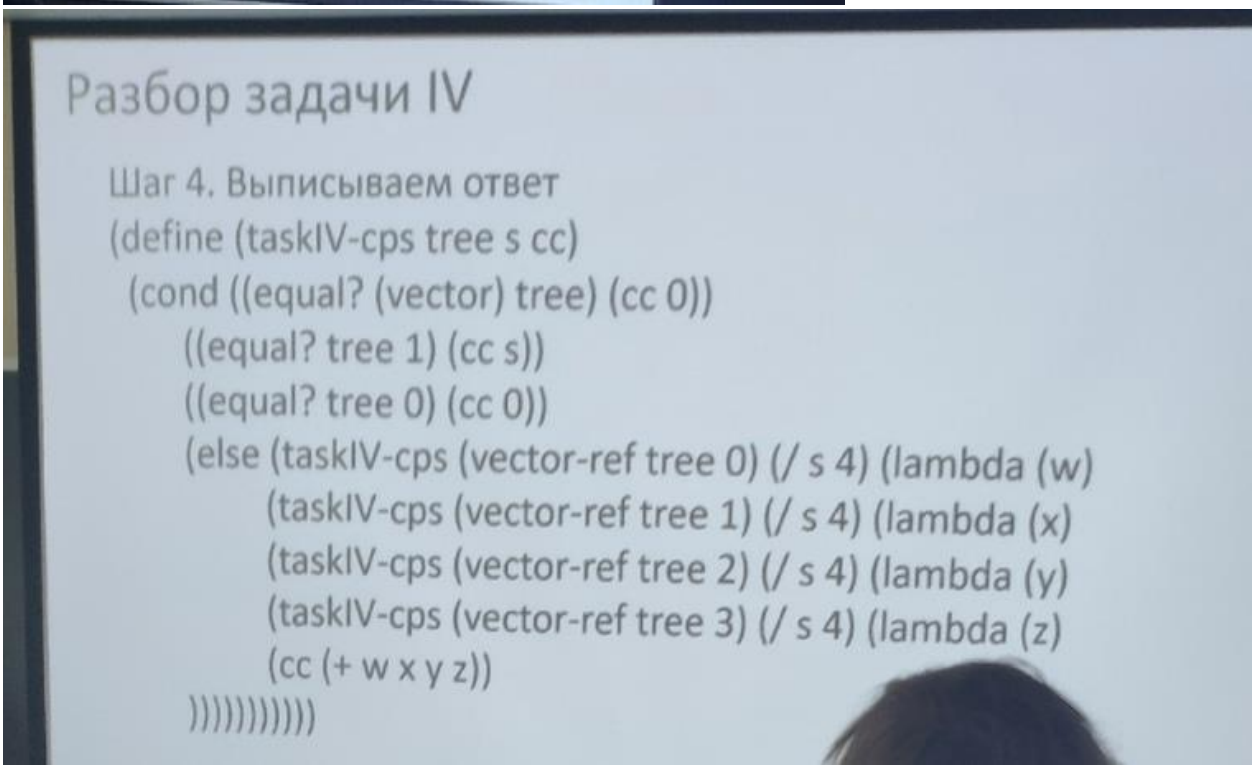
```
(define (taskII tree s)
 (cond ((tree-empty? tree) 0)
       ((equal? tree 1)  s)
       ((equal? tree 0)  0)
       (else (+ (taskII (vector-ref tree 0) (/ s 4))
                (taskII (vector-ref tree 1) (/ s 4))
                (taskII (vector-ref tree 2) (/ s 4))
                (taskII (vector-ref tree 3) (/ s 4))
       )))
)
```




Разбор задачи IV
Шаг 2. Применим cc ко всем веткам cond
(define (taskIV-cps tree s cc)
(cond ((equal? (vector) tree) (cc 0))
((equal? tree 1) (cc s))
((equal? tree 0) (cc 0))
(else (cc (+ (taskIV-cps (vector-ref tree 0) (/ s 4))
(taskIV-cps (vector-ref tree 1) (/ s 4))
(taskIV-cps (vector-ref tree 2) (/ s 4))
(taskIV-cps (vector-ref tree 3) (/ s 4))
))))
)

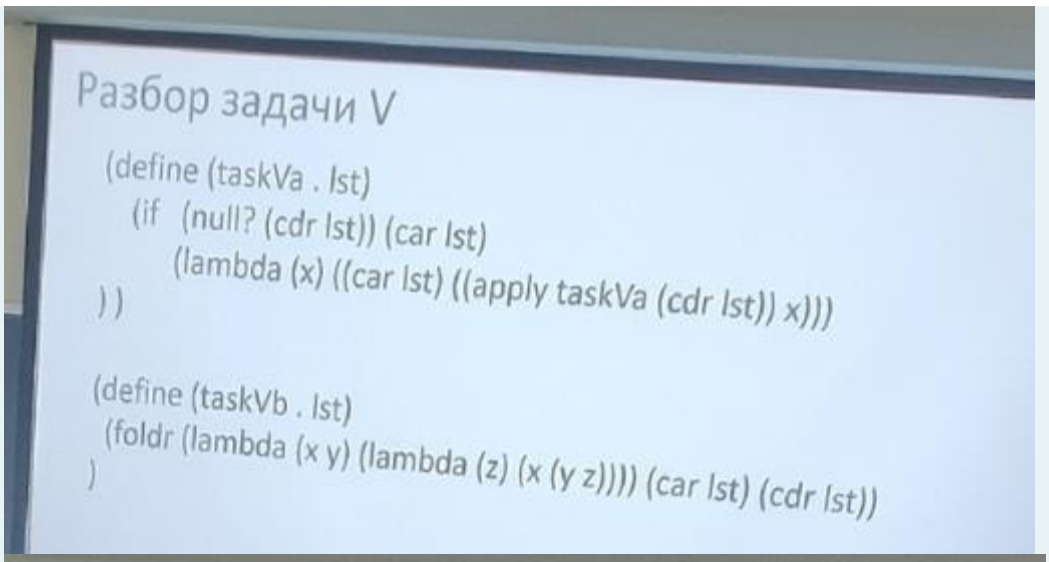

Разбор задачи IV
Шаг 3.а. Борем рекурсивные вызовы в else-ветке
(else (cc (+ (taskIV-cps (vector-ref tree 0) (/ s 4))
(taskIV-cps (vector-ref tree 1) (/ s 4))
(taskIV-cps (vector-ref tree 2) (/ s 4))
(taskIV-cps (vector-ref tree 3) (/ s 4))
)))
О. в. относительно первого вызова:
(lambda (w) (cc (+ w
(taskIV-cps (vector-ref tree 1) (/ s 4))
(taskIV-cps (vector-ref tree 2) (/ s 4))
(taskIV-cps (vector-ref tree 3) (/ s 4))
)))

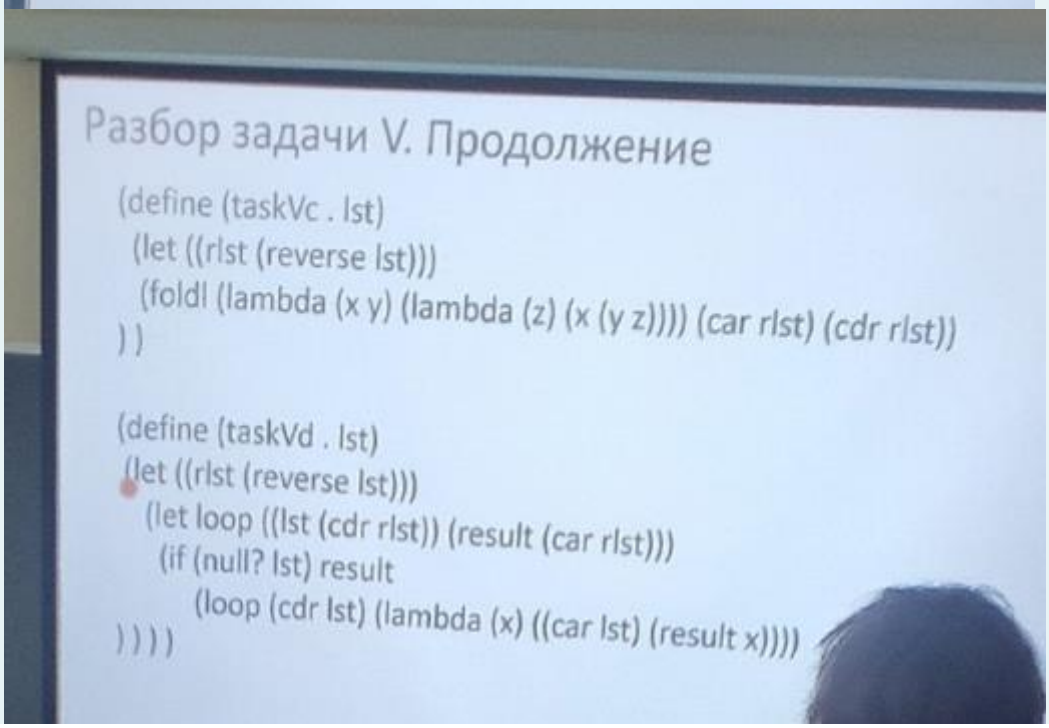




Демо11-5

V. Функция **(taskV f1 f2 ... fn)** возвращает в качестве результата функцию, являющуюся суперпозицией функций от одного аргумента **f1**, **f2**, ... **fn** (**n**>0 натуральное) так что **((taskV f1 f2 ... fn-1 fn) x)** сначала вычислит результат применения **fn** к **x**, затем **fn-1** к полученному результату (**fn x)**, и т. д. до **f1**. Реализуйте функцию **taskV**, считая, что хотя бы 1 аргумент обязательно должен быть в любом её вызове. Не используйте готовые функции вроде **compose1** и т. п..





Демо12-1 и Демо12-3 это чисто теория

Составьте на листе тексты ответов и нарисуйте стрелочную диаграмму. Сфотографируйте и загрузите имидж или pdf-версию. Удачное решение принесёт до 3 баллов (максимум 1 балл за 1 вопрос). Составляемые Вами примеры не должны совпадать или мало отличаться от примеров из книг, со слайдов лекций. Так как проверка производится вручную, то предоставляется лишь одна попытка.

I. Как работает специальная форма **let** (обычная, не именованная)? Приведите пример Scheme-кода с её осмысленным, оправданным прагматически использованием.

III. Расскажите об аппликативном порядке вычислений в подстановочной модели. В чём он заключается? Каковы его преимущества и недостатки по сравнению с нормальным порядком (точнее с тем порядком, который авторы учебника назвали

"нормальным")? Проиллюстрируйте свой ответ примерами Scheme-кода и пояснениями.

Демо12-2

II. Нарисуйте общую стрелочную диаграмму для значений **a** и **b**, исходя из определений, вычисленных друг за другом в интерпретаторе:

> (define a (list (list 1 2) (cons (cons 3 4) 5)))
> (define b (list (cdr a) (car a) 1))
> (define c (cons (cdr b) b))