Тест начат	суббота, 19 апреля 2025, 16:31
Состояние	Завершены
Завершен	суббота, 19 апреля 2025, 17:45
Прошло времени	1 ч. 14 мин.
Оценка	Еще не оценено

Вопрос **1**

Выполнен

Балл: 25,00

Составьте код решений задач в редакторе или от руки на листе бумаги по своему выбору. Код из редактора объедините в общем txt-файле. Лист сфотографируйте или отсканируйте. Вставьте имиджи в общий PDF-файл. При вставке верно сориентируйте имиджи. При необходимости заведите несколько страниц в PDF-файле.
В решениях задач **I-V** *нельзя* применять присваивание, мутаторы, перевод списков в вектора или наоборот. Решения, в которых не будут соблюдены эти ограничения, будут оценены ниже максимума. Эффективность решений задач по времени счёта и памяти учитывается. Правильное, но неэффективное решение может быть оценено ниже максимальных баллов. Для реализации итеративного процесса используйте именованный **let**, а не вспомогательную функцию. Если в решении Вы определяете собственные дополнительные функции, то **обязательны** комментарии на **русском** языке, поясняющие назначение и работу этих функций. Код Ваших решений должен быть **читаемым**.

I. (≤5 баллов) При помощи свёртки реализуйте функцию (adjacent-map fun lst). Функция принимает список (e1 e2 ... en) длиной n, n>1. Функция возвращает список из результатов применения fun к парам соседних элементов исходного списка ((fun e1 e2) (fun e2 e3) ... (fun en-1 en)). Не используйте другие функции высших порядков для списков: ormap, filter, map и т. п.. Запрещено преобразовывать список во вспомогательную структуру данных (вектор, множество, хэш или др).
Примеры:
> (adjacent-map * '(1 2 3 4 5 6)) => (2 6 12 20 30)
> (adjacent-map < '(1 2 1 3 4 3)) => (#t #f #t #t #f)

II. (≤5 баллов) Опишите функцию (fun-ii n), строящую дерево рекурсивных вызовов при вычислении n-го числа Фибоначчи "наивным способом" по рекуррентным формулам F(0)=0, F(1)=1, F(i)=F(i-1)+F(i-2). Для n=0 или n=1 дерево вызовов состоит из одной корневой вершины, при которой записано число F(0) или F(1), соответственно. Для n=2 дерево вызовов состоит из корня, при котором записано число F(2), с левым поддеревом являющимся деревом вызовов для n=1, и правым поддеревом, являющимся деревом вызовов для n=0. Для n=i дерево вызовов состоит из корня, при котором записано число F(i), с левым поддеревом являющимся деревом вызовов для n=i-1, и правым поддеревом, являющимся деревом вызовов для n=i-2. Для внутреннего представления дерева используйте вектора --
(vector <целое_число_при_корне> <поддерево1> <поддерево2>), где количество поддеревьев равно двум, и пустые поддеревья задаются явно пустыми векторами. Примеры:
> (fun-ii 0) => #(0 #() #())
> (fun-ii 1) => #(1 #() #())
> (fun-ii 2) => #(1 #(1 #() #()) #(0 #() #()))
Указание: Перед построением дерева рассчитайте и сохраните в векторе все необходимые числа Фибоначчи, а затем используйте вектор в роли мемоизирующей таблицы.

III. (≤5 баллов) Опишите функцию (fun-iii n tree), принимающую неотрицательное целое n и двоичное дерево в векторном представлении. Функция проверяет, является ли tree деревом вызовов при вычислении n-го числа Фибоначчи "наивным способом" (см. II). Примеры:
> (fun-iii 0 #(0 #() #())) => #t
> (fun-iii 0 #(1 #() #())) => #f
> (fun-iii 2 #(1 #(1 #() #()) #(0 #() #()))) => #t
Указание: Перед построением дерева рассчитайте и сохраните в векторе все необходимые числа Фибоначчи, а затем используйте вектор в роли мемоизирующей таблицы.

IV. (≤5 баллов) Реализуйте (fun-iv-cps n tree cc) версию функции (fun-iii n tree), переписанную в стиле передачи продолжений, так, что в ней используется только хвостовая рекурсия Примеры:
> (fun-iv-cps 0 #(0 #() #()) not) => #f
> (fun-iv-cps 0 #(1 #() #()) not) => #t
> (fun-iv-cps 2 #(1 #(1 #() #()) #(0 #() #())) not) => #f

V. (≤5 баллов) Реализуйте (fun-v n tree) версию функции (fun-iii n tree), переписанную с использованием нелокального возврата для эффективной работы в ситуациях, когда досрочно, до завершения обхода всего дерева становится известно, что результат #f. Примеры:
> (fun-v 0 #(0 #() #())) => #t
> (fun-v 0 #(1 #() #())) => #f
> (fun-v 2 #(1 #(1 #() #()) #(0 #() #()))) => #t

solution1.txt

Вопрос **2**

Выполнен

Баллов: 1,00 из 1,00

Пусть даны определения:

```
(define a (list 'list 1 2 '(list)))
(define b '(list 1 2 (list)))
(define c (list* (car b) '1 (cddr a)))
```

Укажите те и только те утверждения, которые верны.

- [ ] a. Списки a и b совпадают (являются одним и тем же объектом).
- [ ] b. Списки (cddr a) и (cddr b) совпадают (являются одним и тем же объектом).
- [x] c. Списки a и b одинаково выглядят, но не совпадают (не являются одним и тем же объектом).
- [ ] d. Списки (cdr a) и (cdr c) совпадают (являются одним и тем же объектом).
- [x] e. Списки a и c одинаково выглядят, но не совпадают (не являются одним и тем же объектом).
- [x] f. Списки (cddr a) и (cddr b) одинаково выглядят, но не совпадают (не являются одним и тем же объектом).
- [ ] g. Списки a и c совпадают (являются одним и тем же объектом).
- [x] h. Списки (cdr a) и (cdr c) одинаково выглядят, но не совпадают (не являются одним и тем же объектом).

Вопрос **3**

Выполнен

Баллов: 1,00 из 1,00

Пусть дан код:

```
(define p (mcons -20 20))
(define (func p) (set! p (mcons (mcdr p) (mcdr p))))
(func p)
```

В этом коде `mcons`, `mcar`, `mcdr` - конструктор и селекторы мутируемых пар.
Укажите те и только те утверждения, которые будут верны после выполнения этого кода, переданного интерпретатору в окне определений.

- [x] a. После выполнения кода хвост пары p будет равен 20.
- [ ] b. После выполнения кода хвост пары p будет равен -20.
- [ ] c. После выполнения кода голова пары p будет равна 20.
- [ ] d. После выполнения кода голова и хвост пары p станут равны друг другу.
- [x] e. После выполнения кода пара p не изменится, т. к. функция меняет только свой параметр, и эти изменения не затрагивают p, описанное вне определения функции.
- [x] f. Функция func ничего не возвращает.
- [x] g. После выполнения кода голова пары p будет равна -20.
- [ ] h. Выполнение кода приведёт к ошибке, так как параметр функции не может быть мутируемой парой.
- [ ] i. Выполнение кода приведёт к ошибке, так как нельзя в вызове функции как аргумент указывать переменную, имя которой совпадает с именем параметра функции.

Вопрос **4**

Выполнен

Баллов: 1,00 из 1,00

Не используя среду программирования на Scheme, укажите верные и только верные утверждения о выражении:
(let ((x 3) (t expt)) (lambda (n) ((lambda (x f) (f x n)) x t)))

- [ ] a. результатом вычисления выражения будет список
- [x] b. результатом вычисления выражения будет функция от одного аргумента
- [ ] c. результатом вычисления выражения будет функция от двух аргументов
- [ ] d. результатом вычисления выражения будет спецформа lambda
- [ ] e. результатом вычисления выражения будет спецформа let
- [ ] f. в выражении ошибка

Вопрос **5**

Выполнен

Баллов: 0,75 из 1,00

Укажите те и только те варианты, в которых по правилам языка Scheme записаны числа.

- [ ] a. 10e3/5
- [ ] b. #o678
- [ ] c. 1/0
- [ ] d. i
- [x] e. #o567
- [x] f. 0/1
- [x] g. +1i
- [ ] h. 3/5e10

Вопрос **6**

Нет ответа

Балл: 10,00

Указание: Запишите ответы на все вопросы на листе бумаги, отсканируйте или сфотографируйте. Либо набейте текст и создайте иллюстрации в редакторе. Вставьте имиджи в единый PDF-файл. Или составьте из отредактированных текстов и иллюстраций единый PDF-файл. Верно сориентируйте имиджи в PDF. При необходимости заведите несколько страниц в PDF-файле. Созданный файл загрузите в MOODLE. В ответах на текстовые вопросы засчитываются *только Ваши собственные осмысленные оригинальные примеры*, существенно отличающиеся от рассмотренных в лекциях и в книгах. Составляя ответы явно напишите о том, что спрашивается в *каждой части* вопроса. Избегайте *тавтологий* (истинных, но не несущих смысла утверждений).

3.I. (≤6 баллов) Расскажите о стиле программирования с потоками. Что такое поток и каковы базовые операции с ним? Какие преимущества даёт использование потоков в программах? Какие недостатки связаны с использованием потоков? Поясните свои ответы кодом примера, в котором решается осмысленная задача и демонстрируются указанные Вами преимущества и/или недостатки.

3.II. (≤4 балла) Расскажите об именованном let. Чем полезна эта конструкция? Зачем она введена, ведь есть же обычный let? Поясните свои ответы кодом примера, в котором решается осмысленная задача и подкрепляются приведённые в Вашем ответе аргументы.

Вопрос **7**

Выполнен

Баллов: 1,00 из 1,00

Укажите те и только те утверждения, которые верны.

- [x] a. Спецформа let* является "синтаксическим сахаром" так как может быть заменена использованием спецформы lambda.
- [ ] b. Спецформа let* сначала вычисляет значения введённых с её помощью локальных имен, а затем вычисляет выражение, являющееся её телом. При этом порядок вычисления локальных имён стандартом языка не определён и зависит от реализации.
- [ ] c. Спецформа lambda является "синтаксическим сахаром" так как может быть заменена использованием спецформы let*.
- [x] d. Спецформа let* сначала вычисляет значения введённых с её помощью локальных имен, а затем вычисляет выражение, являющееся её телом. При этом порядок вычисления локальных имён стандартом языка определён -- слева направо, и вычисленные значения предшествующих локальных имён будут учтены при вычислении последующих локальных имён.

Вопрос **8**

Выполнен

Баллов: 0,00 из 1,00

Укажите те и только те утверждения о подстановочной модели вычислений, которые верны.
Указание: под "нормальным порядком" здесь всюду имеется в виду тот порядок, который указан под таким названием в учебнике.

- [ ] a. при нормальном порядке вычислений комбинации сначала вычисляются все её подвыражения, а потом результаты их вычислений используются, чтобы найти значение всей комбинации
- [x] b. при аппликативном порядке вычислений комбинации сначала вычисляются все её подвыражения, а потом результаты их вычислений используются, чтобы найти значение всей комбинации
- [ ] c. в подстановочной модели при вычислении комбинации независимо от порядка вычислений сначала вычисляются все её подвыражения, а потом результаты их вычислений используются, чтобы найти значение всей комбинации
- [x] d. в подстановочной модели при вычислении любой комбинации выполняется подстановка в тело вызываемой функции либо вычисленных значений аргументов комбинации, либо выражений, указанных как аргументы комбинации
- [ ] e. в подстановочной модели при вычислении одних комбинаций выполняется подстановка, а при вычислении других -- не выполняется.

Вопрос **9**

Выполнен

Баллов: 1,00 из 1,00

Укажите те и только те утверждения об условной спецформе if, которые верны.

- [x] a. Допускается только такое использование спецформы if, при котором указывается три аргумента -- предикат, следствие и альтернатива.
- [x] b. При использовании спецформы if, если значение первого аргумента не равно #f, то вычисляется второй аргумент и его значение будет результатом вычисления if.
- [ ] c. Допускается только такое использование спецформы if, при котором первый аргумент является выражением, возвращающим либо #f, либо #t.
- [ ] d. При использовании спецформы if вычисляются только два её аргумента: либо первый и второй; либо второй и третий.
- [ ] e. Допускается сокращённое использование спецформы if, при котором указывается только два аргумента -- предикат и следствие.
- [x] f. При использовании спецформы if вычисляются только два её аргумента: либо первый и второй; либо первый и третий.
- [x] g. При использовании спецформы if, если значение первого аргумента равно #f, то вычисляется третий аргумент и его значение будет результатом вычисления if.
- [ ] h. Иногда при использовании спецформы if вычисляются все три её аргумента, но не всегда.

Вопрос **10**

Выполнен

Баллов: 1,00 из 1,00

Укажите те и только те утверждения, которые верны.

- [ ] a. Функция может быть определена так, что её можно будет вызывать с различными количествами аргументов. Для этого нужно в одном и том же окружении завести несколько определений одноимённых функций с нужными количествами параметров.
- [ ] b. Функция может быть определена так, что её можно будет вызывать с различными количествами аргументов. Для этого нужно в определении функции (в define) использовать нотацию точечной пары. Для анонимных функций, задаваемых через lambda, подобной возможности нет.
- [x] c. Функция может быть определена так, что её можно будет вызывать с различными количествами аргументов. Для этого нужно в определении функции (в define) использовать нотацию точечной пары. Подобная возможность есть и для lambda.
- [ ] d. Функция не может быть определена так, что её можно будет вызывать с различными количествами аргументов. Подобная возможность есть только у спецформ и у предопределённых функций.
- [ ] e. Функция не может быть определена так, что её можно будет вызывать с различными количествами аргументов. Подобная возможность есть только у спецформ.

Вопрос **11**

Выполнен

Баллов: 1,00 из 1,00

Укажите верные и только верные утверждения про выражение `(define is0? (lambda x (apply = (cons 0 x))))` в модели вычислений с окружениями.

- [x] a. При вычислении этого выражения в первый кадр текущего окружения будет добавлено новое связывание для `is0?`.
- [ ] b. При вычислении этого выражения в текущее окружение будет добавлен новый кадр для хранения связываний `x`.
- [x] c. После вычисления этого выражения можно в текущем окружении вызвать функцию `is0?`. При этом её вызов может содержать произвольное количество аргументов.
- [ ] d. До вычисления этого выражения ни в каком из кадров текущего окружения не допускается наличие связываний для `is0?`.
- [x] e. Это выражение записано по правилам языка Scheme и будет вычислено без ошибок.
- [ ] f. Результат вычисления этого выражения равен будет тому значению, которое получит `is0?`.
- [ ] g. По правилам языка Scheme `is0?` не может являться именем.

Вопрос **12**

Выполнен

Баллов: 1,00 из 1,00

Какое время счёта потребуется в реализованном на Scheme эффективном нахождении суммы **n** первых положительных кратных пяти целых чисел. Составляя ответ, делайте оценки в зависимости от **n**. Базовыми операциями для подсчёта сложности считайте арифметические действия с целыми числами: **+**, **-**, *****, **/**. Представьте себе решение, которое, на Ваш взгляд, достаточно эффективно по счёту, и укажите его наилучшие сложностные характеристики.

- [ ] a. полиномиальное время счёта
- [ ] b. экспоненциальное время счёта
- [x] c. константное время счёта
- [ ] d. линейное время счёта
- [ ] e. логарифмическое время счёта

◄ Видео по восьмой теме (2021)

Перейти на...

Демовариант к/р весна 2025 г. ►