Тест по Функционално програмиране

спец. Компютърни науки, зимен семестър 2011/2012 г.

Данни на студента		
Име:		
Специалност:	**Група:**	**Ф.н.:**

Задача 1. Посочете каква е разликата между `let`, `let*` и `letrec` в Scheme.

Задача 2. Обяснете накратко каква е разликата между `eq?`, `eqv?` и `equal?` в Scheme.

Задача 3. Нарисувайте как би изглеждала глобалната среда E, ако в нея се изпълни кодът даден в колоната **А**. След това нарисувайте как ще се промени средата, след като изпълним дадения в **Б** код.

А	**Б**
`(define x (cons 1 2))` `(define (f y) (set-car! y (cons 5 6)))`	`(f x)` `(set! x 100)`

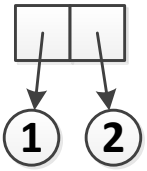
Задача 4. Напишете едно обръщение към функцията на Scheme (accumulate op term null-value a next b), което да генерира списъка:

$$\left(2^0 \quad 2^1 \quad 2^2 \quad \ldots \quad 2^{200}\right)$$

Задача 5. Нека е даден списък `L=( (1) (2 3) (4 (5)) )`.
Използвайки само процедурите `car` и `cdr` и техните производни функции (напр. `cddr`, `cadr`, `cdaddr` и т.н.), напишете изрази, чиято оценка да бъде:

Оценка	**Израз**
A) 1	
B) 2	
C) 3	
D) 4	
E) 5	
F) ‘()	

Задача 6. Графично с двойна правоъгълна кутия изобразяваме наредена двойка, а с кръг – атом. Например `(cons 1 2)` се представя така, както е показано вдясно.



Като използвате тази нотация, нарисувайте как биха изглеждали обектите, които се получават в резултат на оценката на следните изрази в Scheme:

```
A. (list (list 1) (list 2))
B. (cons (list 1 2) (list 1 2))
C. (let ((y 20)) (cons y y))
D. (list '())
```

Задача 7. Посочете каква ще бъде оценката на дадените по-долу изрази на Scheme. Ако смятате, че в някой израз има грешка и той не може да се оцени, посочете каква е тя.

A)	`(eq? (cons 1 (cons 2 (list 3)))`
	`  '(1 2 3)`
	`)`
	Оценка:

B)	`(list? (cons 1 (cons 2 (list 3)))  )` **Оценка:**

C)	`(member (cons 1 2)`
	`  (list (cons 5 6) "Abc" (cons 1 2))`
	`)`
	Оценка:

D)	`((lambda (a) (list a (+ a a))) 4)` **Оценка:**
E)	`(list? (car (list '() 1 2 3 4 5)))` **Оценка:**

F)	`(map (lambda (x) (* x x))`
	`  (list 1 2 3 4 5))`
	Оценка:

G)	`(append (map caddr '((1 2 3) (4 5 6) (7 8 9)))`
	`  (map cdddr '((1 2 3) (4 5 6) (7 8 9)))`
	`)`
	Оценка:

H)	`(let ( (a (list 'a 'b 'c))`
	`  (b (cons 'x a))`
	`  (c (cons 'y b)) )`
	`  (list a b c)`
	`)`
	Оценка:

I)	`(let* ((x 10)`
	`(y 20))`
	`(let* ((x y)`
	`(y x))`
	`(+ x y)`
	`))`
	Оценка:

Задача 8. Напишете кода на функцията filter-stream:

```
(define (filter-stream pred? str)









)
```

Задача 9. Посочете каква ще бъде оценката на дадените по-долу изрази на Хаскел. Ако смятате, че в някой израз има грешка и той не може да се оцени, посочете каква е тя.

A)	`([[3,2,1], [5], [4]] !! 2 ) ++ 3:2:[1]` **Оценка:**
B)	`map (takeWhile (\x-> x < 5)) [[1..], [2..], [3..]]` **Оценка:**
C)	`take 10 (zipWith ($) (map (+) [0..]) [1,1..])` **Оценка:**

D)	`let x = 1`
	`  in let x = x`
	`  in x`
	Оценка:

E)	`dropWhile (\x-> x < 10) [1,3..]` **Оценка:**
F)	`(\a b-> [x | x <- a, x` `` `elem` `` `b]) [1..10] [5..15]` **Оценка:**

Задача 10. Нека предположим, че с помощта на дадените по-долу изрази сме проверили типовете на различни обекти в Хаскел. Какъв е отговор сме получили за всеки от тях?

```
A. :t "Abc"
B. :t zip
C. :t map
D. :t (++)
```