Практика 8. Scheme

материалы преподавателя

Вопросы

Вопрос 1

Что выведет Scheme?

```
scm> (define a (+ 1 2))
```

Ответ

```
a
```

```
scm> a
```

Ответ

```
3
```

```
scm> (define b (+ (* 3 3) (* 4 4)))
```

Ответ

```
b
```

```
scm> (+ a b)
```

Ответ

```
28
```

```
scm> (= (modulo 10 3) (quotient 5 3))
```

Ответ

```
#t
```

```
scm> (even? (+ (- (* 5 4) 3) 2))
```

Ответ

```
#f
```

Вопрос 2

Что выведет Scheme?

```
scm> (if (or #t (/ 1 0)) 1 (/ 1 0))
```

Ответ

```
1
```

```
scm> (if (> 4 3) (+ 1 2 3 4) (+ 3 4 (* 3 2)))
```

Ответ

```
10
```

```
scm> ((if (< 4 3) + -) 4 100)
```

Ответ

```
-96
```

```
scm> (if 0 1 2)
```

Ответ

```
1
```

Вопрос 3

Напиши процедуру, возвращающую факториал числа.

```
(define (factorial x)
```

Ответ

```
    (if (< x 2)
        1
        (* x (factorial (- x 1)))))
```

Вопрос 4

Напиши процедуру, возвращающую n-ое число Фибоначчи.

```
(define (fib n)
```

Ответ

```
    (if (<= n 1)
        n
        (+ (fib (- n 1)) (fib (- n 2))))
```

```
scm> (fib 0)
0
scm> (fib 1)
1
scm> (fib 10)
55
```

Вопрос 5

Создай процедуру, принимающую два списка и соединяющую их.

Простой вызов `(cons a b)` не сработает, потому что результат будет глубоким, а не плоским списком. Вызывать встроенную процедуру `append` нельзя.

```
(define (my-append a b)
```

Ответ

```
    (if (null? a)
        b
        (cons (car a) (my-append (cdr a) b))))
```

```
scm> (my-append '(1 2 3) '(2 3 4))
(1 2 3 2 3 4)
```

Вопрос 6

Напиши процедуру, которая принимает элемент `x`, неотрицательное целое `n` и возвращает список из элементов `x`, повторённых `n` раз.

```
(define (replicate x n)
```

Ответ

```
    (if (= n 0)
        nil
        (cons x (replicate x (- n 1)))))
```

```
scm> (replicate 5 3)
(5 5 5)
```

Вопрос 7

Кодирование длин серий — это метод сжатия, например, последовательностей букв. Список `(a a a b a a a a)` может быть сжат в `((a 3) (b 1) (a 4))`. Эта версия последовательности сохраняет не только сами символы, но и количество их повторений.

Напиши процедуру, которая принимает сжатую последовательность и разжимает её в исходную.

Используй `concat` и `replicate`.

```
(define (uncompress s)
```

Ответ

```
    (if (null? s)
        s
        (concat (replicate (car (car s)) (car (cdr (car s))))
            (uncompress (cdr s)))))
```

```
scm> (uncompress '((a 1) (b 2) (c 3)))
(a b b c c c)
```

Вопрос 8

Напиши процедуру, которая принимает процедуру `fn` и применяет её к каждому элементу списка `lst`.

```
(define (map fn lst)
```

Ответ

```
    (if (null? lst)
        nil
        (cons (fn (car lst)) (map fn (cdr lst)))))
```

```
scm> (map (lambda (x) (* x x)) '(1 2 3))
(1 4 9)
```

Вопрос 9

Дополни следующий код, чтобы получить абстрактный тип данных *дерево*.

```
(define (make-tree label branches) (cons label branches))
```

Ответ

```
(define (label tree) (car tree))
(define (branches tree) (cdr tree))
```

Вопрос 10

Используя АТД из предыдущего вопроса, напиши процедуру, которая вычисляет сумму всех значений в дереве. Считай, что там целые числа.

Неплохо было бы использовать процедуру `map`, также может пригодиться вспомогательная функция для вычисления суммы элементов списка.

```
(define (tree-sum tree)
```

Ответ

```
(define (tree-sum tree)
    (+ (label tree) (sum (map tree-sum (branches tree)))))

(define (sum lst)
    (if (null? lst) 0 (+ (car lst) (sum (cdr lst)))))
```