Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЕТ

по лабораторной работе №1

по курсу «Функциональное и Логическое программирование»

на тему: «Списки в Lisp. Использование стандартных функций»

Студент ИУ7-63Б _____ Лагутин Д. В.

(Группа) (Подпись, дата) (Фамилия И. О.)

Преподаватель _____ Толпинская Н. Б.

(Подпись, дата) (Фамилия И. О.)

Москва, 2023 г.

Теоретические вопросы

1. Элементы языка: определение, синтаксис, представление в памяти

Вся информация (данные и программы) в Lisp представляется в виде символьных выражений — S-выражений.

По определению

S-выражение ::= <атом>|<точечная пара>.

Элементарные значения структур данных:

— Атомы:

  – символы (идентификаторы) — синтаксически – набор литер (букв и цифр), начинающихся с буквы,
  – специальные символы — T, Nil (используются для обозначения логических констант),
  – самоопределимые атомы — натуральные числа, дробные числа (например 2/3), вещественные числа, строки – последовательность символов, заключенных в двойные апострофы (например “abc”).

— Структуры:

Точечные пары ::=(<атом>.<атом>)|(<атом>.<точечная пара>)|
|(<точечная пара>.<атом>)|
|(<точечная пара>.<точечная пара>),

Список ::= <пустой список>|<непустой список>, где

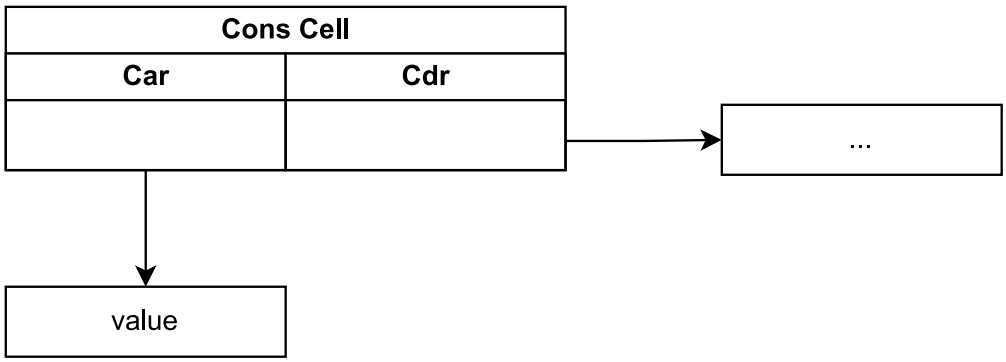
<пусой сисок> ::= ()|Nil,

<непустой список> ::= (<первый элемент>.<хвост>),

<первый элемент> ::= <S-выражение>,

<хвост> ::= <список>.

Любая непустая структура Lisp в памяти представляется списковой ячейкой, хранящей два указателя: на голову (первый элемент) `car` и хвост `cdr` — все остальное.



2. Особенности языка Lisp. Структура программы. Символ апостроф

Lisp - интерпретируемый язык символьной обработки. Вся информация (данные и программы) представляется в виде S-выражений, это даёт возможность выдать программу за данные и заставить её менять саму себя. По умолчанию список считается вычислимой формой, в которой первый элемент — название функции, остальные — аргументы функции.

В основе языка Lisp лежит λ-исчисление, согласно которому, любые вычислительные выражения могут быть преобразованы в вид функций от 1-го аргумента.

Поскольку программа и данные представлены списками, то их нужно как-то различать. Для этого была создана функция `quote`, блокирующая вычисления, а ’ — ее сокращенное обозначение.

Таким образом, символ апострофа ’ — функциональная блокировка, эквивалентен функции quote. Блокирует вычисление выражения. Таким образом, выражение воспринимается интерпретатором как данные.

3. Базис языка Lisp. Ядро языка

Базис — минимальный набор инструментов и структкр данных языка, который позволяет реализовать любую поставленную задачи.

Базис языка представлен:

— атомами;

— структурами;

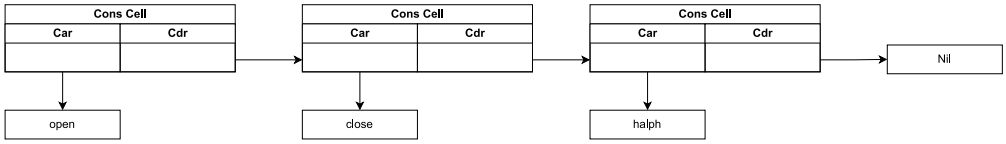
— функциями

`atom, eq, cons, car, cdr, cond, quote, lambda, eval, label.`

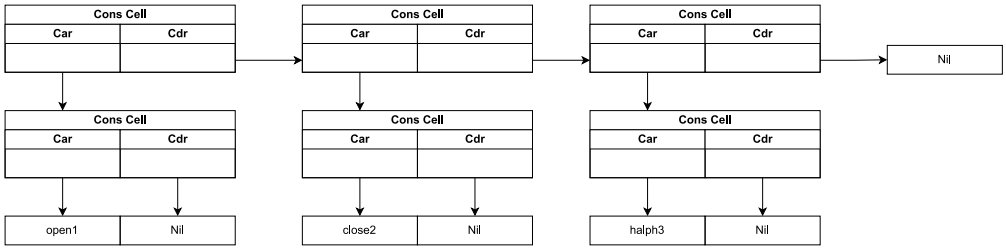
1 Задание

Представить следующие списки в виде списочные ячеек:

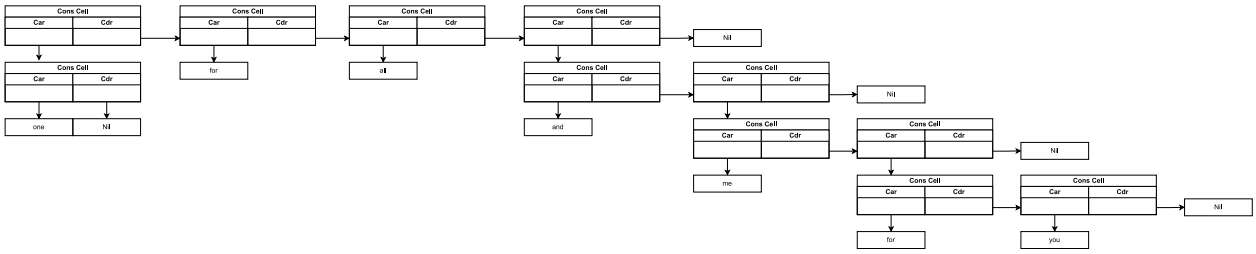
```
'(open close halph)
```



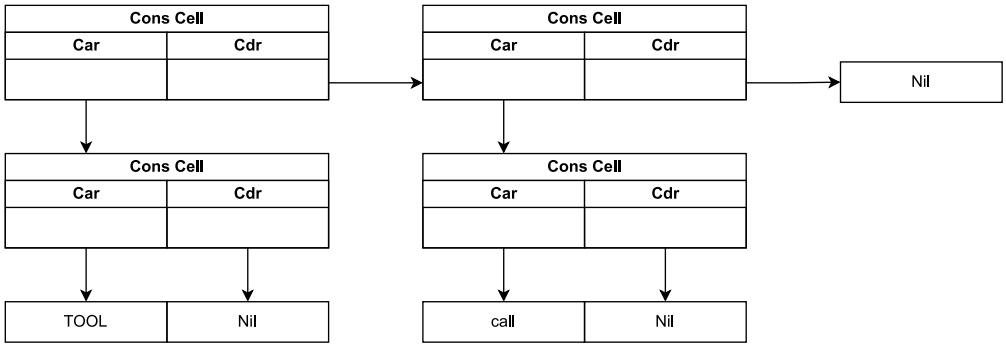

```
'((open1) (close2) (halph3))
```



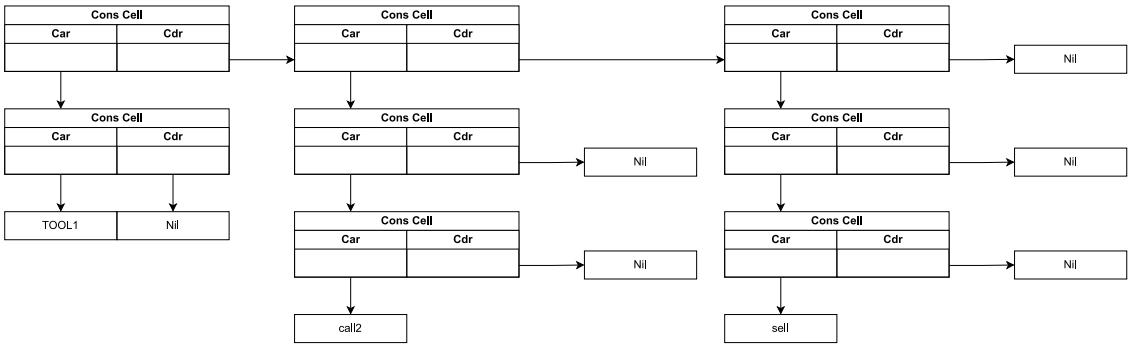

```
'((one) for all (and (me (for you))))
```



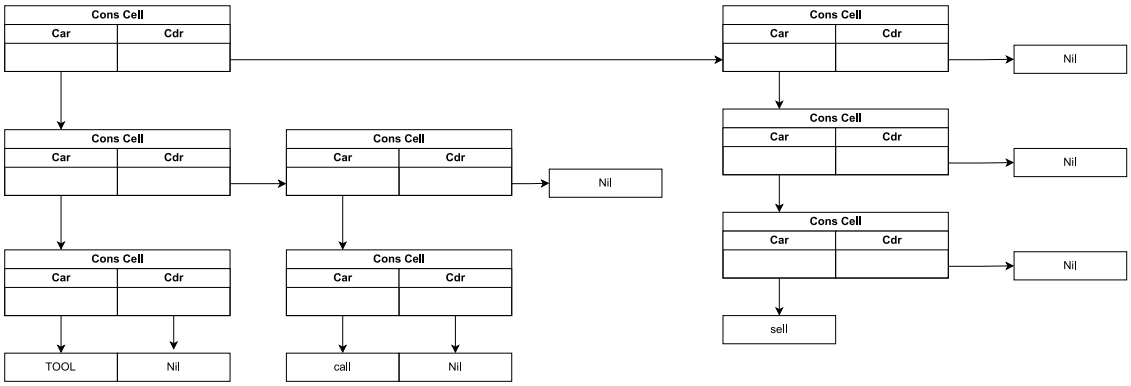

```
'((TOOL) (call))
```



```
'((TOOL1) ((call2)) ((sell)))
```




```
'(((TOOL) (call)) ((sell)))
```



2 Задание

Используя только функции CAR и CDR, написать выражения, возвращающие 1) второй 2) третий 3) четвертый элементы заданного списка.

```
1 ;;; Второй элемент списка
2 (car (cdr '(a b c d)))
3
4 ;;; Третий элемент списка
5 (car (cdr (cdr '(a b c d))))
6
7 ;;; Четвертый элемент списка
8 (car (cdr (cdr (cdr '(a b c d)))))
```

3 Задание

Что будет в результате вычисления выражений?

```
(CAADR '((blue cube) (red pyramid)))
;;; red

(CADR '((abc) (def) (ghi)))
;;; (def)

(CDAR '((abc) (def) (ghi)))
;;; Nil

(CADDR '((abc) (def) (ghi)))
;;; (ghi)
```

4 Задание

Напишите результат вычисления выражений и объясните как он получен:

```
1  (list 'Fred 'and 'Wilma)
2  ;;; (FRED AND WILMA)
3  ;;; Функция list создает список из переданнаых элементов
4
5  (list 'Fred '(and Wilma))
6  ;;; (FRED (AND WILMA))
7  ;;; Функция list создает список из переданнаых элементов, второй элемемнт -
8  ;;; список из x2 элементов
9
10 (cons Nil Nil)
11 ;;; (Nil)
12 ;;; Функция cons создает списковуй ячейку из переданных аргументов.
13 ;;; Таким образом, создается ячейка указатель cdr которой равен nil -
14 ;;; признак конца списка, а значение cda - nil
15
16 (cons T Nil)
17 ;;; (T)
18
19 (cons Nil T)
20 ;;; (NIL . T)
21 ;;; Второй элемент - не список - создается точечные пара
22
23 (list Nil)
24 ;;; (NIL)
25 ;;; Список из элемента nil
26
27 (cons '(T) Nil)
28 ;;; ((T))
29 ;;; Список, первый элемент которого - список
30
31 (list '(one two) '(free temp))
32 ;;; ((ONE TWO) (FREE TEMP))
33 ;;; Список, оба элемента которого - списки
34
35 (cons 'Fred '(and Wilma))
36 ;;; (FRED AND WILMA)
37 ;;; Хвост списка - список, поэтому полученная ячейка - список по определению
38
39 (cons 'Fred '(Wilma))
40 ;;; (FRED WILMA)
41
42
43
```

```
(list Nil Nil)
;;; (NIL NIL)
;;; Список из двух элементов nil

(list T Nil)
;;; (T NIL)

(list Nil T)
;;; (NIL T)

(cons T (list Nil))
;;; (T NIL)

(list '(T) Nil)
;;; ((T) NIL)

(cons '(one two) '(free temp))
;;; ((ONE TWO) FREE TEMP)
```

5 Задание

Написать лямбда-выражение и соответствующую функцию, представить результаты в виде списочных ячеек.

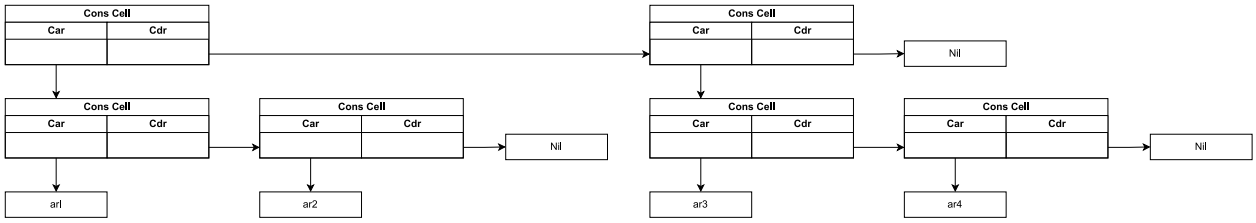
Написать функцию `(f arl ar2 ar3 ar4)`, возвращающую список: `((arl ar2) (ar3 ar4))`.

```
((lambda (arl ar2 ar3 ar4)
   (cons (cons arl (cons ar2 nil)) (cons (cons ar3 (cons ar4 nil)) nil)))
 1 2 3 4)

(defun f1-cons (arl ar2 ar3 ar4)
  (cons (cons arl (cons ar2 nil)) (cons (cons ar3 (cons ar4 nil)) nil)))

((lambda (arl ar2 ar3 ar4) (list (list arl ar2) (list ar3 ar4))) 1 2 3 4)

(defun f1-list (arl ar2 ar3 ar4)
  (list (list arl ar2) (list ar3 ar4)))
```



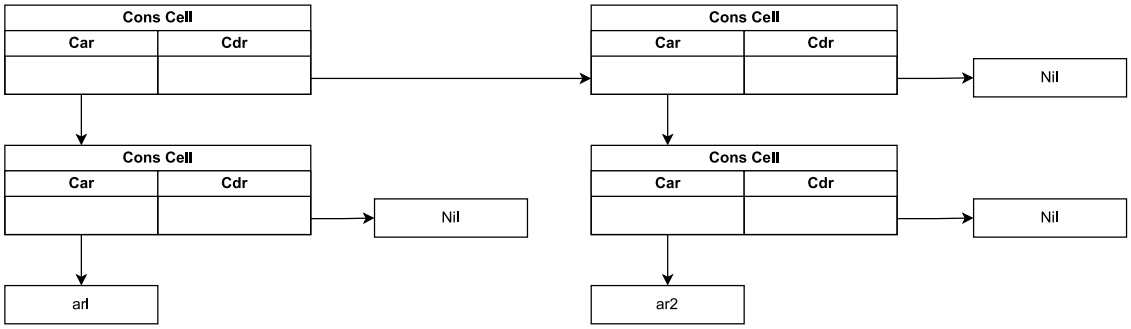
Написать функцию `(f arl ar2)`, возвращающую `((arl) (ar2))`.

```
((lambda (arl ar2)
  (cons (cons arl nil) (cons (cons ar2 nil) nil))) 1 2)

(defun f2-cons (arl ar2)
  (cons (cons arl nil) (cons (cons ar2 nil) nil)))

((lambda (arl ar2)
   (list (list arl) (list ar2))) 1 2)

(defun f2-list (arl ar2)
  (list (list arl) (list ar2)))
```



Написать функцию `(f arl)`, возвращающую `(((arl)))`.

```
((lambda (arl) (cons (cons (cons arl nil) nil) nil)) 1)

(defun f3-cons (arl) (cons (cons (cons arl nil) nil) nil))

((lambda (arl) (list (list (list arl)))) 1)

(defun f3-list (arl) (list (list (list arl))))
```