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»

Отчёт
по лабораторной работе №1

Название: Списки в LISP. Использование стандартных функций.

Дисциплина: Функциональое и логическое программирование

Студент	ИУ7-64Б		Л.Е.Тартыков
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель			Н.Б.Толпинская
		(Подпись, дата)	(И.О. Фамилия)
Преподаватель			Ю.В.Строганов
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2022

1 Теоретические вопросы

1.1 Элементы языка: определение, синтаксис, представление в памяти

К элементарным значениям структур данных относятся.

1. **Атомы**:
   - символы. Используются как идентификаторы любого объекта. Синтаксически представляются буквами, цифрами; начинаются с буквы;
   - специальные символы — T, Nil. Используются для обозначения логических констант;
   - самоопределимые атомы — натуральные, дробные или вещественные числа, строки, заключенные в двойные кавычки;
2. **Точечные пары**. Строятся при помощи бинарных узлов (узлы разделены точкой). Определяются как на листинге 1.1.

Листинг 1.1 – Определение точечной пары.

```
1  Точечные пары ::= (<атом>.<атом>) |
2                    (<атом>.<точечная пара>) |
3                    (<точечная пара>.<атом>) |
4                    (<точечная пара>.<точечная пара>)
```

Синтаксическое представление отражено на листинге 1.2.

Листинг 1.2 – Определение точечной пары.

```
1      (A.B)
```

3. **Список**. Строятся при помощи бинарных узлов. Определяются как на листинге 1.3.

Листинг 1.3 – Определение списка.

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

<пустой список> ::= () | Nil,
<непустой список> ::= (<первый элемент>, <хвост>),
<первый элемент> ::= <S-выражение>,
<хвост> ::= <список>.
```

Синтаксически список представляется как представлено на листинге 1.4.

Листинг 1.4 – Представление списка.

```
Nil
()

(A. (B. (C. (D. ()))))
(A B C D)
(A (B C) (D (E)))
```

Любая непустая структура Lisp в памяти представляется списковой ячейкой, хранящей два указателя: на голову и хвост.

Универсальным разделителем между атомами является пробел.

1.2 Особенности языка Lisp. Структура программы. Символ апостроф.

Вся информация (данные и программа) в Lisp представляется в виде символьных выражений — S-выражений. Определение представлено на листинге 1.5.

Листинг 1.5 – Определение S-выражения.

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

Символ апостроф является упрощенным способом обращения к функции quote. Пример использования функции quote и символа апостроф приведен на листинге 1.6.

Листинг 1.6 – Пример использования функции quote и символа апостроф.

```
1 (quote (ATOM B)) => (ATOM B)
2 '(ATOM B) => (ATOM B)
```

1.3 Базис языка Lisp. Ядро языка.

Базис язык Lisp образуют: атомы, структуры, базовые функции, базовые функционалы. Базовый набор функций включает 8 встроенных:

- **car** — возвращает голову списка, являющегося значением её единственного аргумента;
- **cdr** — возвращает хвост списка, являющего значением её единственного аргумента;
- **cons** — имеет два аргумента (cons e1 e2). Строит новый список, первым элементом которого является значение первого аргумента e1, а хвостом - e2;
- **atom** — возвращается T, если значением её единственного аргумента является атом (числовой или символьный); иначе - Nil;
- **eq** – проверяет совпадение двух своих аргументов-атомов. Возвращается значение T, когда значением одного из аргументов является атом и значения аргументов равны. Иначе возвращается Nil;
- **quote** — в качестве значения возвращает аргумент, не вычисляя его (используется для блокировки вычисления аргумента).
- **eval** — выполняет двойное вычисление своего аргумента. Такой подход может понадобиться либо для снятия блокировки вычислений (установленного функцией quote), либо же для вычисления нового функционального вызова, сформированного в ходе первого вычисления;
- **cond** — условное выражение; служит средством разветвления вычислений. Пример представлен на листинге 1.7.

Листинг 1.7 – Вызов функции cond.

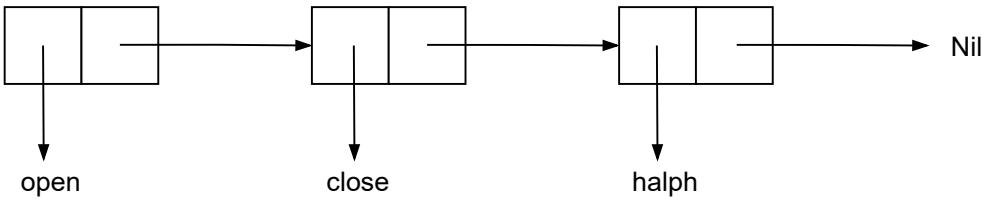
```
1     (cond (p_1 e_1) (p_2 e_2) ... (p_n e_n)), n >= 1
```

Выражения $(p_i e_i)$ являются ветвями условного выражения, а выражения-формы p_i - условиями ветвей.

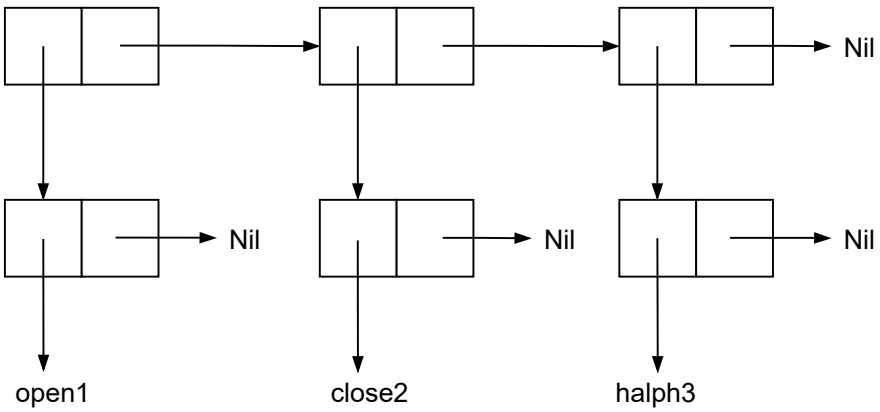
2 Практические задания

2.1 Представить следующие списки в виде списочных ячеек:

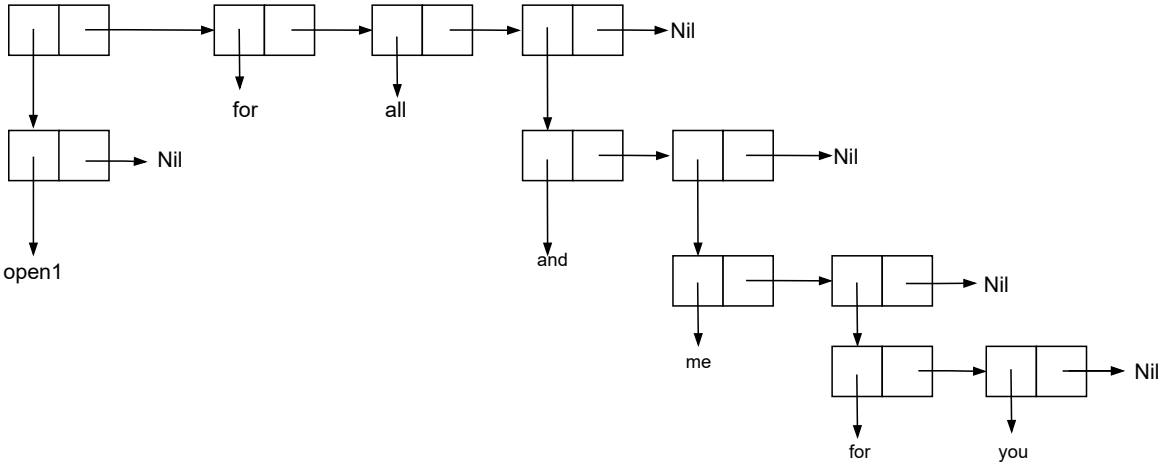
'(open close halph)



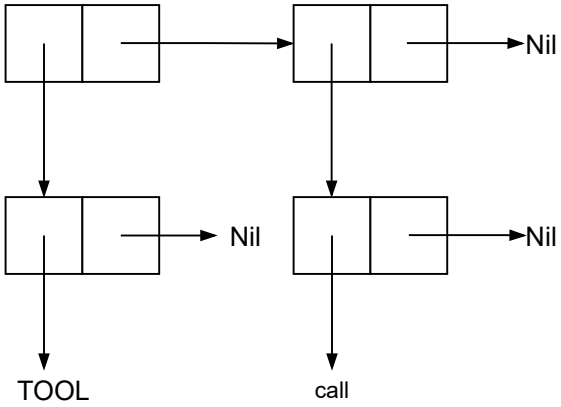
'((open1) (close2) (halph3))



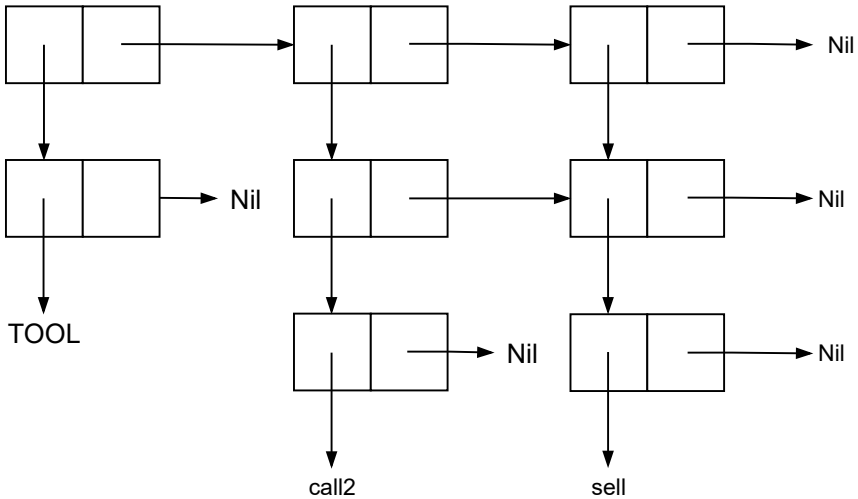
'((one) for all (and (me (for you))))



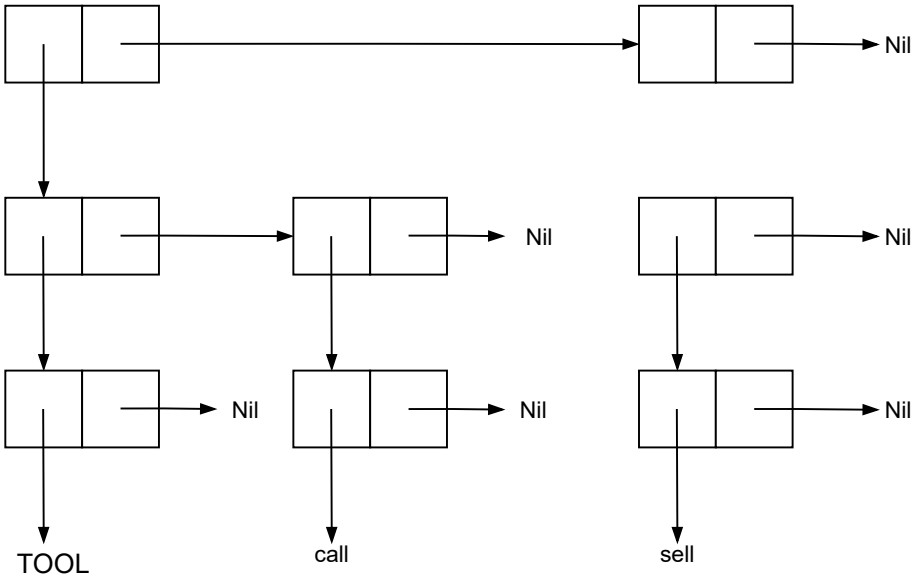
'((TOOL) (call))



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



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



2.2 Использующие только функции CAR и CDR, написать выражения, возвращающие второй, третий, четвертый элементы заданного списка

1. второй элемент: (car (cdr '(a b c d e f)))
2. третий элемент: (car (cdr (cdr '(a b c d e f))))
3. четвертый элемент: (car (cdr (cdr (cdr '(a b c d e f)))))

2.3 Что будет в результате вычисления выражений?

a) (CAADR '((blue cube) (red pyramid)))
(car (car (cdr '((blue cube) (red pyramid)))))
(car (car '(red pyramid)))

(car '(red)) => red

b) (CDAR '((abc) (def) (ghi)))

(cdr (car '((abc) (def) (ghi))))

(cdr '((abc)) => Nil

c) (CADR '((abc) (def) (ghi)))

(car (cdr '((abc) (def) (ghi))))

(car '((def) (ghi))) => (def)

d) (CADDR '((abc) (def) (ghi)))

(car (cdr (cdr '((abc) (def) (ghi)))))

(car (cdr '((def) (ghi))))

(car '((ghi))) => (ghi)

2.4 Напишите результат вычисления выражений и объясните, как он получен

1. (list 'Fred 'and 'Wilma)

   (cons 'Fred (cons 'and (cons 'Wilma ())))

   (Fred and Wilma)

2. (list 'Fred '(and Wilma))

   (cons 'Fred (cons '(and Wilma) ()))

   (cons 'Fred (and Wilma))

   (Fred (and Wilma))

3. (cons Nil Nil) => (Nil)

4. (cons T Nil) => (T)

5. (cons Nil T) => (Nil T)

6. (list Nil)

   (cons Nil Nil) => (Nil)

7. (cons '(T) Nil) => ((T))

8. (list '(one two) '(free temp))

   (cons '(one two) (cons '(free temp)))

   ((one two) (free temp))

9. (cons 'Fred '(and Wilma))

   (Fred and Wilma)

10. (cons 'Fred '(Wilma))

    (Fred Wilma)

11. (list Nil Nil)

    (cons Nil Nil) => (Nil Nil)

12. (list T Nil)

    (cons T Nil) => (T Nil)

13. (list Nil T)

    (cons Nil T) => (Nil T)

14. (cons T (list Nil))

    (cons T (cons Nil Nil))

    (cons T Nil Nil)

    (T Nil Nil)

    (T Nil)

15. (list '(T) Nil)

    (cons '(T) (cons Nil Nil))

    (cons '(T) Nil)

    ((T) Nil)

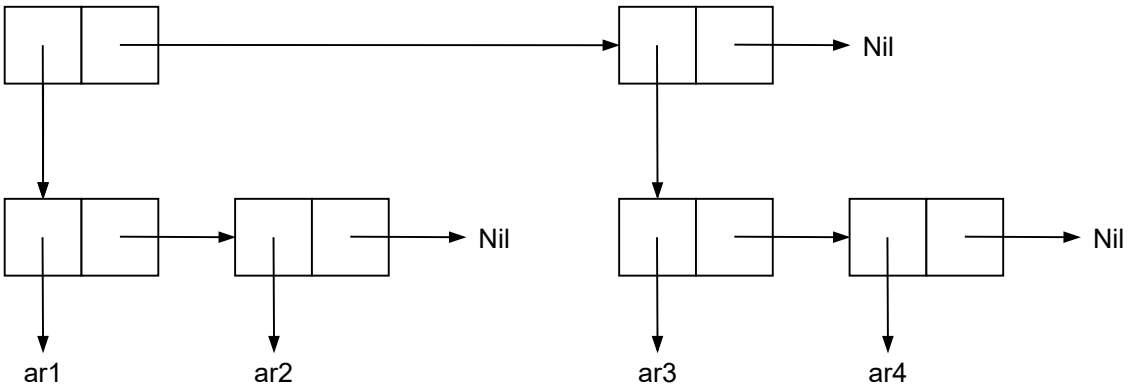
16. (cons '(one two) '(free temp))

    ((one two) free temp)

2.5 Написать лямбда-выражение и соответствующую функцию. Представить результаты в виде списочных ячеек.

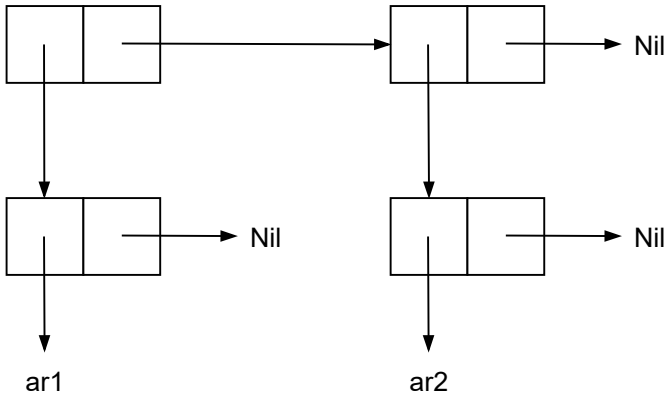
- Написать функцию (f ar1 ar2 ar3 ar4), возвращающую список ((ar1 ar2) (ar3 ar4)).

  (defun f (ar1 ar2 ar3 ar4) (cons '(ar1 ar2) (cons '(ar3 ar4) ())))



- Написать функцию (f ar1 ar2), возвращающую ((ar1) (ar2)).

  (defun f (ar1 ar2) (cons '(ar1) (cons '(ar2) ())))



- Написать функцию (f ar1), возвращающую (((ar1))).

  (defun f (ar1) (cons (cons '(ar1) ()) ()))