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 дисциплине «Функциональное и логическое программирование»

Тема Списки в Lispe. Использование стандартных функций.

Студент Зайцева А. А.

Группа ИУ7-62Б

Оценка (баллы) _____

Преподаватели Толпинская Н.Б., Строганов Ю. В.

Москва — 2022 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>.

Элементы языка Lisp включают в себя:

- 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, вещественные числа, строки – последовательность символов, заключенных в двойные апострофы (например, “abc”);
- Более сложные данные – списки и точечные пары (структуры), которые строятся с помощью унифицированных структур – блоков памяти – бинарных узлов. Определения:

  Точечная пара ::= (<атом> . <атом>) | (<атом> . <точечная пара>) | (<точечная пара> . <атом>) | (<точечная пара> . <точечная пара>);

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

  <пустой список> ::= () | Nil,

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

  <первый элемент> ::= <S-выражение>,

  <хвост> ::= <список>.

  Список – это частный случай S-выражения.

Синтаксически:

любая структура (точечная пара или список) заключается в круглые скобки: (A . B) – точечная пара, (A) – список из одного элемента. Пустой список изображается как Nil или ();

непустой список по определению может быть изображен: (A . (B . (C . (D . ()))))), допустимо изображение списка последовательностью атомов, разделенных пробелами – (A B C D).

Элементы списка могут быть списками (любой список заключается в круглые скобки), например – (A (B C) (D C)). Таким образом, синтаксически наличие скобок является признаком структуры – списка или точечной пары.

Любая непустая структура Lisp в памяти представляется списковой ячейкой, хранящей два указателя: на голову (первый элемент) и хвост – всё остальное. Точечная пара в памяти представляется бинарным узлом.

2. Особенности языка Lisp. Структура программы. Символ апостроф.

Отличительные особенности Lisp: только символьная обработка; все можно представить в виде функций.

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

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

В зависимости от контекста одни и те же объекты могут играть роль переменных или констант, причем значения и того, и другого могут быть произвольной сложности. Если объект играет роль константы, то для объявления константы достаточно заблокировать его вычисление, то есть как бы взять его в кавычки, отмечающие буквально используемые фразы, не требующие обработки.

Апостроф – сокращённое обозначение функции quote.

quote - блокирует вычисление своего аргумента. В качестве своего значения выдаёт сам аргумент, не вычисляя его. Перед константами - числами и атомами T, Nil можно не ставить апостроф.

3. Базис языка Lisp. Ядро языка.

Базис – это минимальный набор инструментов языка и стркутур данных, который позволяет решить любые задачи.

Базис Lisp :

- атомы и структуры (представляющиеся бинарными узлами);

- базовые (несколько) функций и функционалов: встроенные — примитивные функции (atom, eq, cons, car, cdr); специальные функции и функционалы (quote, cond, lambda, eval, apply, funcall).

Функцией называется правило, по которому каждому значению одного или нескольких аргументов ставится в соответствие конкретное значение результата.

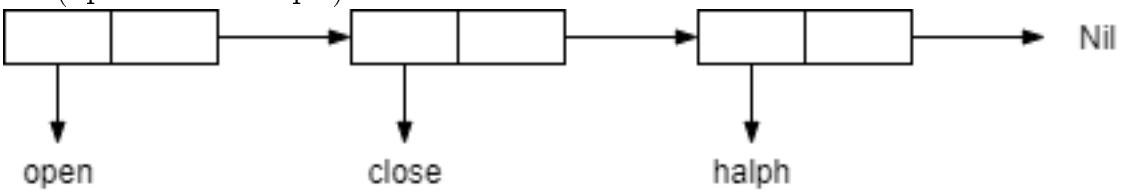
Функционалом, или функцией высшего порядка называется функция, аргументом или результатом которой является другая функция.

Некоторые функции системы необходимо реализовывать в виде машинных подпрограмм, они образуют ядро системы. Ядро - основные действия, которые наиболее часто используются. Понятие «ядро» шире, чем понятие «базис».

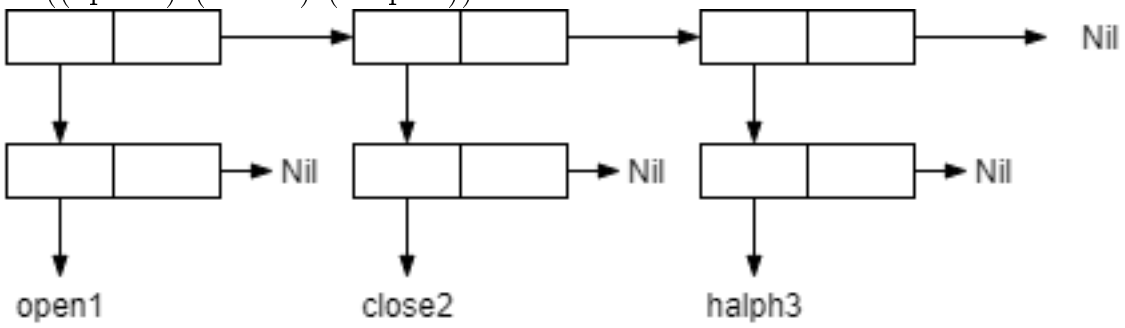
Практические задания

1. Представить следующие списки в виде списочных ячеек

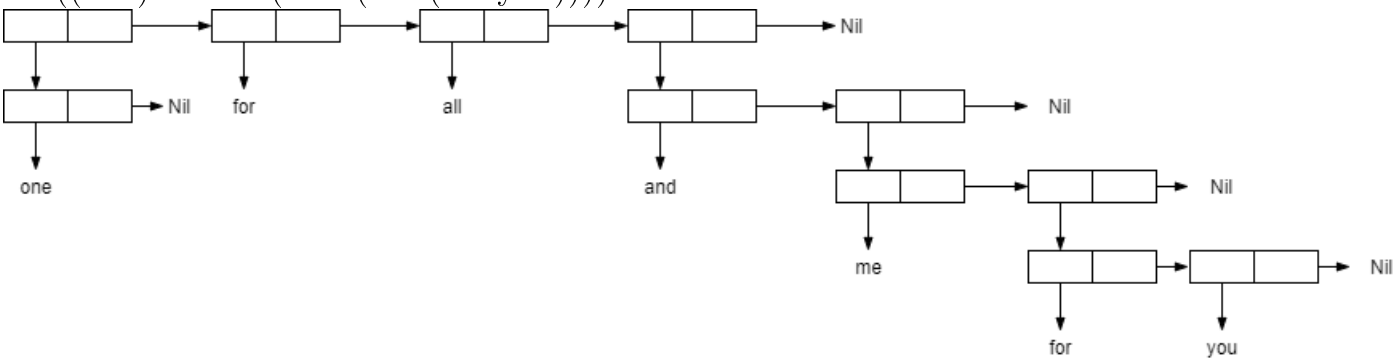
1. '(open close halph)



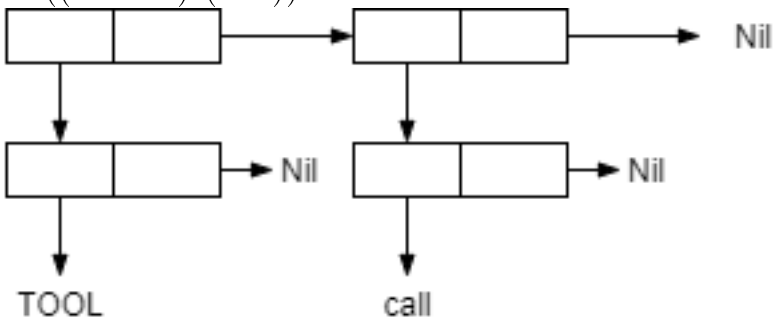
2. '((open1) (close2) (halph3))



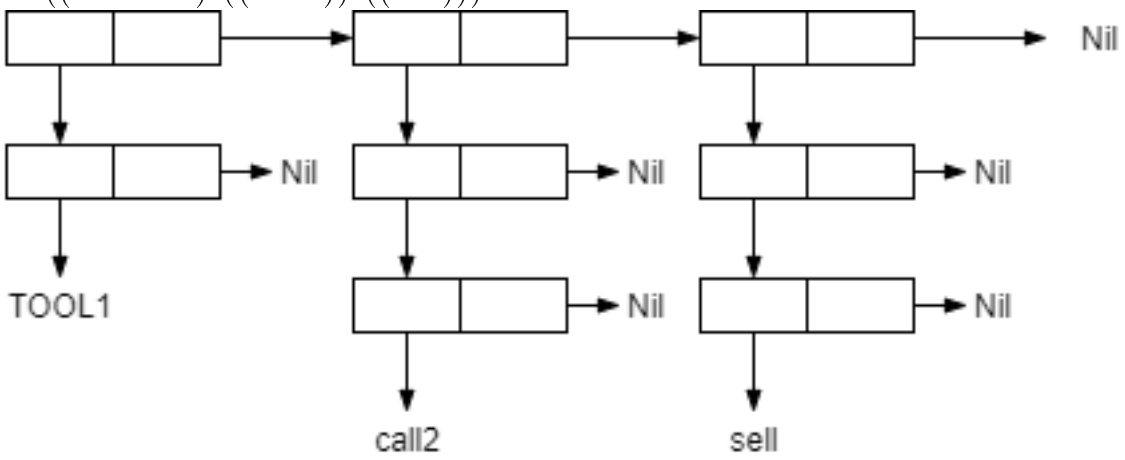
3. '((one) for all (and (me (for you))))



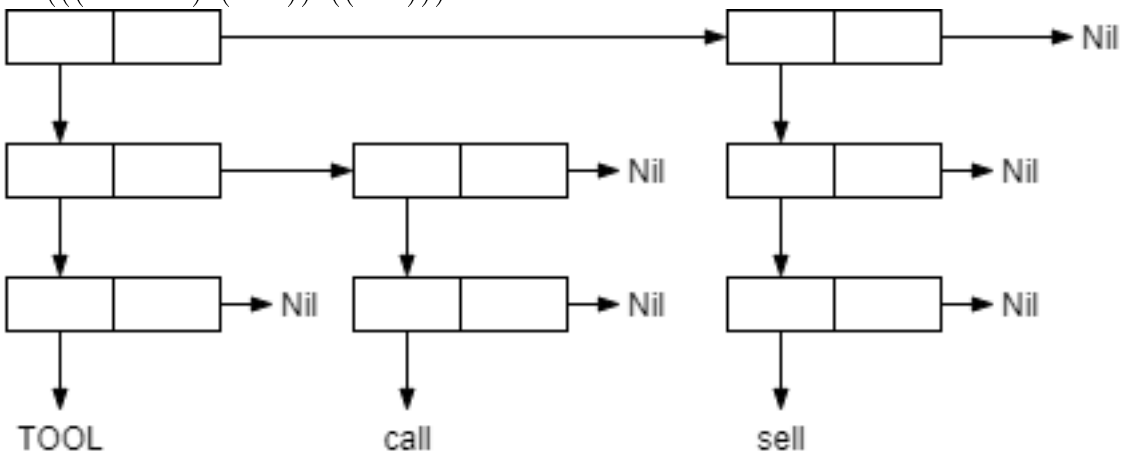
4. '((TOOL) (call))



5. '((TOOL1) ((call2)) ((sell)))



6. '(((TOOL) (call)) ((sell)))



2. Используя только функции CAR и CDR, написать выражения, возвращающие:

1. второй

```
(CAR (CDR '(1 2 3 4 5))) => 2

(CADR '(1 2 3 4 5)) => 2
```

2. третий

```
(CAR (CDR (CDR '(1 2 3 4 5)))) => 3

(CADDR '(1 2 3 4 5)) => 3
```

3. четвертый элементы заданного списка

```
(CAR (CDR (CDR (CDR '(1 2 3 4 5))))) => 4

(CADDDR '(1 2 3 4 5)) => 4
```

3. Что будет в результате вычисления выражений?

1. (CAADR '((blue cube) (red pyramid)))

```
(CAADR '((blue cube) (red pyramid))) => red
1. (CDR '((blue cube) (red pyramid))) => ((red pyramid))
2. (CAR '((red pyramid))) => (red pyramid)
3. (CAR '(red pyramid)) => red
```

2. (CDAR '((abc) (def) (ghi))

```
(CDAR '((abc) (def) (ghi))) => Nil
1. (CAR '((abc) (def) (ghi))) => (abc)
2. (CDR '(abc)) => Nil
```

3. (CADR '((abc) (def) (ghi)))

```
(CADR '((abc) (def) (ghi))) => (def)
1. (CDR '((abc) (def) (ghi))) => ((def) (ghi))
2. (CAR '((def) (ghi))) => (def)
```

4. (CADDR '((abc) (def) (ghi)))

```
(CADDR '((abc) (def) (ghi))) => (ghi)
1. (CDR '((abc) (def) (ghi))) => ((def) (ghi))
2. (CDR '((def) (ghi))) => ((ghi))
3. (CAR '((ghi))) => (ghi)
```

4. Напишите результат вычисления выражений и объясните, как он получен

quote - блокирует вычисление своего аргумента. В качестве своего значения выдаёт сам аргумент, не вычисляя его. Перед константами - числами и атомами T, Nil можно не ставить апостроф. Апостроф – сокращённое обозначение функции quote.

list - не имеет фиксированного количества аргументов. Создает и возвращает список, у которого голова - это первый аргумент, хвост - все остальные аргументы.

cons - имеет фиксированное количество аргументов (два). В случае, когда аргументами являются атомы создает точечную пару. В случае, когда первый аргумент – атом, а второй – список, атом становится головой, а второй аргумент (список) становится хвостом.

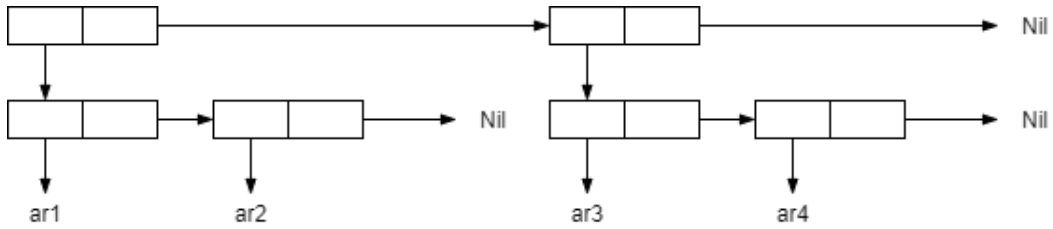
```
(list 'Fred 'and 'Wilma) => (Fred and Wilma).
(list 'Fred '(and Wilma)) => (Fred (and Wilma))
(cons Nil Nil) => (Nil)
(cons T Nil) => (T)
(cons Nil T) => (Nil . T)
(list Nil) => (Nil)
(cons '(T) Nil)  => ((T))
(list '(one two) '(free temp)) => ((one two) (free temp))

(cons 'Fred '(and Willma)) => (Fred and Willma)
(cons 'Fred '(Wilma)) => (Fred Willma)
(list Nil Nil) => (Nil Nil)
(list T Nil) => (T Nil)
(list Nil T) => (Nil T)
(cons T (list Nil)) => (T Nil)
;; (cons T (list Nil)) = (cons T (Nil)) = (T Nil)
(list '(T) Nil) => ((T) Nil)
(cons '(one two) '(free temp)) => ((one two) free temp))
```

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

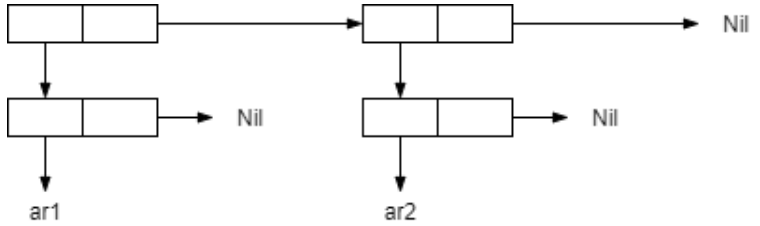
1. Написать функцию (f ar1 ar2 ar3 ar4), возвращающую список: ((ar1 ar2) (ar3 ar4)).

```
(defun f1 (ar1 ar2 ar3 ar4) (list (list ar1 ar2) (list ar3 ar4)))
(f1 1 2 3 4) => ((1 2) (3 4))
```



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

```
(defun f2 (ar1 ar2) (list (list ar1) (list ar2)))
(f2 1 2) => ((1) (2))
```



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

```
(defun f3 (ar1) (list (list (list ar1))))
(f3 1) => (((1)))
```