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 №4

по курсу «Функциональное и логическое программирование»

на тему: «Использование управляющих структур, работа со списками»

Студент ИУ7-63Б _____ Недолужко Д. В.
(Группа) (Подпись, дата) (И. О. Фамилия)

Преподаватель _____ Толпинская Н. Б.
(Подпись, дата) (И. О. Фамилия)

2022 г.

1 Практическая часть

1.1 Чем принципиально отличаются функции cons, list, append?

Пусть:

Листинг 1.1 – объявление функций из условия

```
1         (setf lst1 '(a b))
2         (setf lst2 '(c d))
```

В Таблице 1.1 приведены результаты вычисления выражений.

Таблица 1.1 – Результаты вычисления выражений

Выражение	**Результат**
`(cons lst1 lst2)`	`((A B) C D)`
`(list lst1 lst2)`	`((A B) (C D))`
`(append lst1 lst2)`	`(A B C D)`

1.2 Каковы результаты вычисления следующих выражений?

В Таблице 1.2 приведены результаты вычисления выражений.

Таблица 1.2 – Результаты вычисления выражений

Выражение	**Результат**
`(reverse ())`	`(Nil)`
`(last ())`	`(Nil)`
`(reverse '(a))`	`(a)`
`(last '(a))`	`(a)`
`(reverse '((a b c)))`	`((a b c))`
`(last '((a b c)))`	`((a b c))`

1.3 Написать, по крайней мере, два варианта функции, которая возвращает последний элемент своего списка-аргумента

Листинг 1.2 – Вариант 1

```
(defun mylast1 (lst)
  (cond ((null lst)Nil)
        ((null (cdr lst))lst)
        (T(mylast1 (cdr lst)))))
```

Листинг 1.3 – Вариант 2

```
(defun mylast2 (lst)
  (let ((revlst (reverse lst)))
       (if (null lst)
           Nil
           (car revlst))))
```

1.4 Написать, по крайней мере, два варианта функции, которая возвращает свой список-аргумент без последнего элемента

Листинг 1.4 – Вариант 1

```
(defun removelast1 (lst)
  (cond ((null lst)Nil)
        ((null (cdr lst))Nil)
        (T(cons (car lst) (removelast (cdr lst))))))
```

Листинг 1.5 – Вариант 2

```
(defun removelast2 (lst)
  (and lst (reverse (cdr (reverse lst)))))
```

1.5 Написать простой вариант игры в кости, в котором бросаются две правильные кости

Если сумма выпавших очков равна 7 или 11 – выигрыш, если выпало (1,1) или (6,6) — игрок имеет право снова бросить кости, во всех остальных случаях ход переходит ко второму игроку, но запоминается сумма выпавших очков. Если второй игрок не выигрывает абсолютно, то выигрывает тот игрок, у которого больше очков. Результат игры и значения выпавших костей выводить на экран с помощью функции print.

Листинг 1.6 – Игра в кости

```
(defun is_step_win (cube1p cube2p)
  (let ((sum (+ cube1p cube2p)))
       (OR (= 7 sum)(= 11 sum))))

(defun is_step_repeat (cube1p cube2p)
  (AND (= cube1p cube2p) (OR (= 1 cube1p) (= 6 cube1p))))

(defun play_step (player_name)
  (let* ((cube1p (+ (random 8) 1))
         (cube2p (+ (random 8) 1))
         (sum (+ cube1p cube2p)))
        (cond ((is_step_win cube1p cube2p)
               (AND (print `(,player_name - points ,cube1p ,
                  cube2p - ABSOLUTE_WIN))
                    (list T sum)))
              ((is_step_repeat cube1p cube2p)
               (AND (print `(,player_name - points ,cube1p ,
                  cube2p - REPEAT))
                    (let ((res (play_step player_name)))
                         (list (first res)(+ sum (second res)))))
                      )
              (T(AND (print `(,player_name - points ,cube1p ,
                 cube2p - TRANSFER))
                     (list Nil sum))))))

(defun play_game ()
  (let ((resu1 (play_step 'USER1)))
       (if (first resu1)
           (print `(USER1 is a winner))
           (let ((resu2 (play_step 'USER2)))
                (if (first resu2)
                    (print `(USER2 is a winner))
                    (cond ((> (second resu1)(second resu2))
                           (print '(USER1 is a winner)))
                          ((< (second resu1)(second resu2))
                           (print '(USER2 is a winner)))
                          (T(print '(DRAW)))))))))
```

2 Контрольный вопросы

2.1 Базис языка

Базис состоит из:

1. структуры, атомы;
2. встроенные (примитивные) функции (`atom`, `eq`, `cons`, `car`, `cdr`);
3. специальные функции и функционалы, управляющие обработкой структур, представляющих вычислимые выражения (`quote`, `cond`, `lambda`, `label`, `eval`).

2.2 Классификация функций

Функции в `Lisp` классифицируют следующим образом:

- чистые математические функции;
- рекурсивные функции;
- специальные функции — формы (сегодня 2 аргумента, завтра - 5);
- псевдофункции (создают эффект на внешнем устройстве);
- функции с вариативными значениями, из которых выбирается 1;
- функции высших порядков — функционал: используется для синтаксического управления программ (абстракция языка).

По назначению функции разделяются следующим образом:

- конструкторы — создают значение (`cons`, например);
- селекторы — получают доступ по адресу (`car`, `cdr`);
- предикаты — возвращают `Nil`, `T`.
- функции сравнения – такие как: `eq`, `eql`, `equal`, `equalp`.

2.3 Способы создания функций

Функции в Lisp можно задавать следующими способами:

Lambda-выражение

Синтаксис:

(lambda <λ-список> форма)

Пример:

Листинг 2.1 – Функция определенная Lambda-выражением

```
1 (lambda (a b) (sqrt (+ (* a a) (* b b))))
```

Именованая функция

Синтаксис:

(defun <имя функции> <λ-выражение>)

Пример:

Листинг 2.2 – Функция определенная Lambda-выражением

```
1 (defun hyp (a b) (sqrt (+ (* a a) (* b b))))
```

2.4 Работа функций and, or, if, cond

2.4.1 Функция and

Синтаксис:

Листинг 2.3 – функция and

```
1 (and expression-1 expression-2 ... expression-n)
```

Функция возвращает первое expression, результат вычисления которого = Nil. Если все не Nil, то возвращается результат вычисления последнего выражения.

Примеры:

Листинг 2.4 – пример использования `and`

```
(and 1 Nil 2)
```

Результат: `Nil`

Листинг 2.5 – пример использования `and`

```
(and 1 2 3)
```

Результат: 3

2.4.2 Функция `or`

Синтаксис:

Листинг 2.6 – функция `or`

```
(or expression-1 expression-2 ... expression-n)
```

Функция возвращает первое expression, результат вычисления которого не `Nil`. Если все `Nil`, то возвращается `Nil`.

Примеры:

Листинг 2.7 – пример использования `or`

```
(or Nil Nil 2)
```

Результат: 2

Листинг 2.8 – пример использования `or`

```
(or 1 2 3)
```

Результат: 1

2.4.3 Функция `if`

Синтаксис:

Листинг 2.9 – функция `if`

```
(if condition t-expression f-expression)
```

Если вычисленный предикат не `Nil`, то выполняется `t-expression`, иначе - `f-expression`.

Примеры:

Листинг 2.10 – пример использования `if`

```
(if Nil 2 3)
```

Результат: 3

Листинг 2.11 – пример использования `if`

```
(if 0 2 3)
```

Результат: 2

2.4.4 Функция `cond`

Синтаксис:

Листинг 2.12 – Функция `cond`

```
(cond
  (condition-1 expression-1)
  (condition-2 expression-2)
  ...
  (condition-n expression-n))
```

По порядку вычисляются и проверяются на равенство с `Nil` предикаты. Для первого предиката, который не равен `Nil`, вычисляется находящееся с ним в списке выражение и возвращается его значение. Если все предкаты вернут `Nil`, то и `cond` вернет `Nil`.

Примеры:

Листинг 2.13 – Пример использования `cond`

```
(cond (Nil 1) (2 3))
```

Результат: 3

Листинг 2.14 – Пример использования `cond`

```
(cond (Nil 1) (Nil 2))
```

Результат: `Nil`