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 №4 по курсу

«Функциональное и логическое программирование»

Тема Использование управляющих структур, работа со списками.

Студент Сироткина П.Ю.

Группа ИУ7-66Б

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

Москва — 2022 г.

Практические задания

1. Чем принципиально отличаются функции cons, list, append? Пусть (setf lst1 '(a b)) (setf lst2 '(c d)). Каковы результаты вычисления следующих выражений?

```
(setf lst1 '(a b))
(setf lst2 '(c d))
(cons lst1 lst2)      ; ((A B) C D)
(list lst1 lst2)      ; ((A B) (C D))
(append lst1 lst2)    ; (A B C D)
```

2. Каковы результаты вычисления следующих выражений, и почему?

```
(reverse ())          ; NIL
(last ())             ; NIL
(reverse '(a))        ; (A)
(last '(a))           ; (A)
(reverse '((a b c)))  ; ((A B C))
(last '((a b c)))     ; ((A B C))
```

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

```
(defun get_last(lst) (last lst))
```

```
(defun get_last(lst) (car (reverse lst)))
```

```
(defun get_last(lst) (if (cdr lst) (get_last(cdr lst)) (car lst)))
```

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

```
(defun but_last(lst) (reverse (cdr (reverse lst))))
```

```
(defun but_last(lst)
    (if (cdr lst)) (cons (car lst) (del_last (cdr lst))) Nil)
```

5. Написать простой вариант игры в кости.

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 в праве 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.

```
(setf *random-state* (make-random-state t))

(defun roll_dices()
    (list (+ (random 6) 1) (+ (random 6) 1)))

(defun sum_points(dices)
    (+ (first dices) (second dices)))

(defun abs_winnerp(dices)
    (or (= (sum_points dices) 7) (= (sum_points dices) 11)))

(defun rerollp(dices)
    (or (= (sum_points dices) 2) (= (sum_points dices) 12)))

(defun define_winner(fdices sdices)
    (let ((fsum (sum_points fdices))
          (ssum (sum_points sdices)))
           (cond
               ((abs_winnerp fdices) 1)
               ((abs_winnerp sdices) 2)
               (T (cond
                     ((= fsum ssum) 0)
                     ((> fsum ssum) 1)
                     (T 2))))))

(defun play()
    (let* ((fdices (roll_dices))
           (sdices (roll_dices)))

        (print 'Player_1)
        (print fdices)
        (terpri)

        (print 'Check_reroll)

        (if (rerollp fdices)
            (progn (setq fdices (roll_dices))(print 'True.)(print 'Player_1)
               (print fdices) (terpri))
            (progn (print 'False.) (terpri)))

```

```
40          (print 'Player_2)
41          (print sdices)
42          (terpri)
43
44          (let ((res (define_winner fdices sdices)))
45              (cond
46                  ((= res 0) (print 'Draw.))
47                  ((= res 1) (print 'Player_1_WIN))
48                  (T (print 'Player_2_WIN))))
49          (terpri)))
50
51  (play)
```