Задание 1

Чем принципиально отличаются функции cons, list, append?

1) cons – помещает первый аргумент в начало второго;
2) list – создает список, состоящий из аргументов;
3) append – создает список, состоящий из элементов аргументов.

Пусть (setf lst1 '(a b))

(setf lst2 '(c d)).

Каковы результаты вычисления следующих выражений?

1) (cons lstl lst2) -> ((A B) C D)
2) (list lst1 lst2) -> ((A B) (C D))
3) (append lst1 lst2) -> (A B C D)

Задание 2

Каковы результаты вычисления следующих выражений, и почему?

1) (reverse ()) -> NIL
2) (last ()) -> NIL
3) (reverse '(a)) -> (A)
4) (last '(a)) -> (A)
5) (reverse '((a b c))) -> ((A B C))
6) (last '((a b c))) -> ((A B C))

Задание 3

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

Решение

```
(defun last_v1 (x)
  (if (listp x)
      (if (<= (length x) 1)
       (car x)
       (last_1 (cdr x)))
      NIL))

(defun last_v2 (x)
  (cond ((null (listp x)) NIL)
     ((null (cdr x)) (car x))
     (T (last_v2 (cdr x)))))

(defun last_v3 (x)
  (if (or (null (listp x)) (null x))
      NIL
      (nth (- (length x) 1) x)))

(defun last_v4 (x)
  (if (listp x)
      (car (reverse x))
      NIL))
```

Задание 4

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

Решение

```
(defun except_last_v1 (x)
  (if (null (and (listp x) (cdr x)))
      NIL
      (cons (car x) (except_last_v1 (cdr x)))))
```

```
(defun except_last_v2 (x)
```

```
  (if (listp x)
      (reverse (cdr (reverse x)))))

(defun except_last_v3 (x)
  (if (listp x)
      (butlast x)
      NIL))
```

Задание 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.

Решение

```
(setf *random-state* (make-random-state t))
(defvar Player_1 (cons "Player_1" 0))
(defvar Player_2 (cons "Player_2" 0))

(defun throw_bones ()
  (cons (+ 1 (random 6 *random-state*)) (+ 1 (random 6 *random-state*)))
)

(defun get_bones_sum(bones)
  (+ (car bones) (cdr bones))
)
(defun print_bones_throw (player bones)
  (format t "~a throws ~a" (car player) bones)
  (terpri)
)
```

```
(defun print_player_win (player)
  (format t "~S Wins" (car player))
  (terpri)
)

(defvar win_sum (cons 7 11))
(defvar reroll_sum (list (cons 1 1) (cons 6 6)))

(defun my_compare (first second)
  (and (= (car first) (car second)) (= (cdr first) (cdr second)))
)

(defun check_is_reroll (bones)
  (or (my_compare bones (car reroll_sum)) (my_compare bones (cadr reroll_sum)))
)

(defun player_turn (player)
  (let* ((bones (throw_bones)) (bones_sum (get_bones_sum bones)))
    (print_bones_throw player bones)
    (if (or (= bones_sum (car win_sum)) (= bones_sum (cdr win_sum)))
      (or (print_player_win player) -1)
    ;else
      (if (check_is_reroll bones)
        (or (format t "Reroll")
            (terpri)
            (player_turn player)
        )
        ;else
        bones_sum
      )
    )
  )
)
```

```
(defun play()
  (let* ((first_sum (player_turn Player_1)))
    (if (= first_sum -1)
      (list 1)
      ;else
      (let* ((second_sum (player_turn Player_2)))
        (if (= second_sum -1)
          (list 2)
          ;else
          (if (> first_sum second_sum)
            (print_player_win Player_1)
            (print_player_win Player_2)
          )
        )
      )
    )
  )
)

(play)
```