; 6

(defun throw\_dices ()

    (print

    (cons   (random 6 (make-random-state t))

            (random 6 (make-random-state t)))

    )

)

(defun sum\_dices (pair)

    (+ (car pair) (cdr pair))

)

(defun winp (pair)

    (cond

        ( (eql (sum\_dices pair) 7) T)

        ( (eql (sum\_dices pair) 11) T)

    )

)

(defun plusstepp (pair)

    (cond

        (   (eql (car pair) (cdr pair))

            (cond

                ( (eql (car pair) 1) T)

                ( (eql (car pair) 6) T)

            )

        )

    )

)

; параметры: steps - список из точечных пар - бросков игрока

; return точечную пару:

;   car-указатель: T/Nil - выиграл или не выиграл игрок

;   cdr-указатель: список из точечных пар - бросков игрока

(defun oneplayer (steps)

    (cond

        (   (winp (car steps))

            (cons T steps))

        (   (plusstepp (car steps))

            (oneplayer (cons (throw\_dices) steps)))

        (   T

            (cons Nil steps))

    )

)

(defun play () ((lambda (res1 res2)

    (print "player 1")

    (setf res1 (oneplayer (cons (throw\_dices) nil)))

    (cond

        (   (car res1)

            (print "player 1 WIN")

            (cons res1 res2)

        )

        (   T

            ((lambda ()

                (print "player 2")

                (setf res2 (oneplayer (cons (throw\_dices) nil)))

                (cond

                    (   (car res2)

                        (print "player 2 WIN")

                    )

                    (   T

                        ((lambda (s1 s2)

                            (cond   ((> s1 s2) (print "player 1 WIN"))

                                    ((< s1 s2)  (print "player 2 WIN"))

                                    (T  (print "no winner")))

                        ) (sum\_dices (cadr res1)) (sum\_dices (cadr res2)))

                    )

                )

            ))

        )

    )

) () () ) )

; 7 Написать функцию, которая по своему списку-аргументу lst определяет, является ли он палиндромом (то есть равны ли lst и (reverse lst))

(defun palinomp (lst)

    (cond

        (   (equalp (car lst) nil)

            T)

        (   (equalp (cdr lst) nil)

            T)

        (   (equalp (car lst) (car (last lst)))

            (palinomp (cdr (subst nil (last lst) lst))))

    )

)

(defun palinomp2 (lst)

    (equalp lst (reverse lst))

)

; (format t "~a~%" ( palinomp '(2 (2 3) 1 1 (2 3) 2)))

; 8 Напишите свои необходимые функции, которые обрабатывают таблицу из 4-х точечных пар: (страна . столица), и возвращают по стране - столицу, а по столице - страну.

(defun get\_val4 (key tab)

    (cond

        (   (eql (caar tab) key)

            (cdar tab))

        (   T

            (cond

                (   (eql (caadr tab) key)

                    (cdadr tab))

                (   T

                    (cond

                        (   (eql (caaddr tab) key)

                            (cdaddr tab))

                        (   T

                            (cond

                                (   (eql (caaddr (cdr tab)) key)

                                    (cdaddr (cdr tab)))

                            ))

                    ))

            )

        )

    )

)

(defun get\_key4 (val tab)

    (cond

        (   (eql (cdar tab) val)

            (caar tab))

        (   T

            (cond

                (   (eql (cdadr tab) val)

                    (caadr tab))

                (   T

                    (cond

                        (   (eql (cdaddr tab) val)

                            (caaddr tab))

                        (   T

                            (cond

                                (   (eql (cdaddr (cdr tab)) val)

                                    (caaddr (cdr tab)))

                            ))

                    ))

            )

        )

    )

)

; 9 Напишите функцию, которая умножает на заданное число-аргумент первый числовой элемент списка из заданного 3-х элементного списка-аргумента, когда

; а) все элементы списка - числа,

(defun fa (num lst)

    (cond

        (   (numberp (car lst) )

            (cond

                (   (numberp (cadr lst) )

                    (cond

                        (   (numberp (caddr lst) )

                            (setf (car lst) (\* num (car lst)))

                        ))

                ))

        ))

)

; б) элементы списка - любые объекты.

(defun fb (num lst)

    (cond

        (   (numberp (car lst) )

            (setf (car lst) (\* num (car lst)))

        )

        (   (numberp (cadr lst) )

            (setf (cadr lst) (\* num (cadr lst)))

        )

        (   (numberp (caddr lst) )

            (setf (caddr lst) (\* num (caddr lst)))

        )

    ))