|  |
| --- |
| (defun roll\_dice ()  (print (list (+ (random 6) 1) (+ (random 6) 1))) )  (defun check\_abs\_win (dice)  (or (equal 7 dice )  (equal 11 dice) ) )  (defun check\_reroll (dice)  (or (equal dice '(1 1)) (equal dice '(6 6)))  )  (defun check\_state (dice)  (cond ((check\_abs\_win (+ (first dice) (second dice))) "Gamer absolutely win" ) #| абсолютная победа |#  ((check\_reroll dice) (check\_state (roll\_dice) ) ) #| перебрасывание |#  (T (+ (first dice) (second dice))) #| подсчет очков |#  )  )  (defun turn\_gamer()  (let ((dice\_first (roll\_dice)))  (let ((st (check\_state dice\_first)))  (if (numberp st) st (and (print st) nil) ) ))  )  (defun game ()  (print "first throw")  (let ((first\_sum (turn\_gamer)))  (if first\_sum  (and (print "second throw")  (let ((second\_sum (turn\_gamer)))  (if (and first\_sum second\_sum)  (cond ((> first\_sum second\_sum) (print "first win"))  ((< first\_sum second\_sum) (print "second win"))  (T (print "dead heat"))  )  )  )  )  )  )  nil  ) |

|  |
| --- |
| (defun get\_elem (n1 n2 lst)  (nth n1 (nth n2 lst)) )  (defun get\_f\_sec\_go (lst len)  (\* (floor (/ len 2)) (max (get\_elem 1 0 lst) (get\_elem 1 1 lst))) )  (defun get\_f\_sec\_back (lst len) ; все  (let ((half\_len (floor (/ len 2)))  (first\_el (get\_elem 1 0 lst))  (sec\_el (get\_elem 1 1 lst))  )  (let ((result (+ (\* (- half\_len 1) first\_el) (\* (- half\_len 1) sec\_el))) )  (if (evenp len) result (+ result first\_el)) )  )  )  (defun rec (num s lst)  (if (>= num 3)  (rec (- num 2) (+ s (get\_elem 1 (- num 1) lst)) lst) s) )  (defun solve (lst)  (let ((len (length lst)))  (cond ((= len 0) nil)  ((= len 1) (get\_elem 1 0 lst))  (t (+ (get\_f\_sec\_go lst len) (get\_f\_sec\_back lst len) (rec len 0 lst)) )  )  )  )  (defun escape (lst)  (print lst)  (sort lst #'< :key #'cadr)    (let ((result (solve lst)))  (if (eql result nil)  (print "Incoreect list length")  (print result) )  )  ) |