University of Central Florida

Department of Electrical Engineering & Computer Science

COP4020: Programming Languages I
Spring 2014

Programming project #2 (Lisp – Part 2)

Due March 30th, 2014

Submissions:
1) All programs must be submitted using the DrRacket environment.

2) **Submit one file with all the definitions using the following file name:**
COP4020-HW4-last name-first name.rkt

1) All submissions must be tested in and submitted in a form that is compatible with the DrRacket IDE.

2) Part I: 70 Points: The following is a Software Requirements Specification (SRS) for a program to be written in Racket Lisp:

Create a dice game with two six-sided dice. Throws of "7" and "11" are winners. Throws of "2', "3" and "12" are losers.

The program shall define at least the following functions:

- throw-die: randomly generate and return a number between 1 and 6
- throw-dice: call throw-die, and return a list of two numbers, the throw.
- throw-value: call throw-dice, and return the sum of its two numbers
- five "naming" functions: call throw-dice or throw-value and output either a string or a list with the name of that number, with the following associations of number:name
    - 7:natural
    - 11:yo
    - 2: snake-eyes
    - 3: ace-deuce
    - 12:boxcar

- win: either accept throw or call throw-dice and call throw-value and output either a string or a list declaring "you won!"
- lose: either accept throw or call throw-dice and call throw-value and output either a string or a list declaring "you lost!"

The user will enter:
>(dice1)
The program shall output, using the naming, win and lose functions, either a list or a string:
For throws of 1, 2, 4, 5, 6, 8, 9, 10:
'(You threw [die1] and [die2] which makes a point of [throw])
where [die1] and [die2] are the values of those two dice, and [throw] is their sum.
For throws of 7 or 11:
'(You threw [die1] and [die2] which makes [throw] -- [NATURAL/YO] -- You win!)
For throws of 2, 3, 12:
'(You threw [die1] and [die2] which makes [throw] -- [SNAKE-EYES/ACE-DEUCE/BOXCARS] -- You lose.)

The program shall not generate any errors nor any values for a throw besides 1 through 12.

3) Part II: 20 Points: Extend dice1 into a new game, dice2. In this game, the user can determine which throws are winners or losers. The user shall enter:
>(dice2 (list "value1" "value2" . . . "value*n*"))

The program shall:

- randomly generate a throw, using the same functions as dice1, above
- decode values1 through values*n*, determining which potential throws are winners and losers
- determine whether the throw is a winner or loser, using either two new functions named "winners", "losers" or a single function named "results"
- determine the name of the throw, if any
- output either a string or a list in a manner identical to dice1

Include as comments a few sample invocations of dice2, such as:

; (dice2 (list "myval1" "myval2")
; (dice2 (list "myval3" "myval4" "myval5")
; (dice2 (list "myval1" "myval5")

4) Part III. 10 points. For both programs, alternate formats of the output are acceptable, as long as they convey the same information in a manner that is at least as legible and user-friendly as the above. Full points will be rewarded for a graphic display using Racket's visual language (see http://docs.racket-lang.org/quick/index.html).

5) Submit via Webcourses the definitions of the functions for dice1 and dice2, and these sample invocations, in one (1) file with a Racket file extension, using the following file name format: "HW4_LastName_FirstName.rkt", where LastName and FirstName are the names of the student submitting his or her original work.