CS115 Fall 2010

Assignment 2
Due 4:00pm on Tuesday, Oct 12

For this and all subsequent assignments, you are expected to use the design recipe when writing functions from scratch. Do not copy the purpose directly from the assignment description. The purpose should be written in your own words and include reference to the parameter names of your functions. The solutions you submit must be entirely your own work. Do not look up either full or partial solutions on the Internet or in printed sources.

Please read the course Web page for more information on assignment policies and how to organize and submit your work. Be sure to download the interface file from the course Web page and to follow all the instructions listed in the style guide (on the Web page). Specifically, your solutions should be placed in files `a2qY.rkt`, where `Y` is a value from `1` to `4`. For full marks, it is not sufficient to have a correct program. Be sure to follow all the steps of the design recipe, including the definition of constants and helper functions where appropriate.

For all questions, you may assume that the test data will satisfy all conditions stated for consumed data.

Language Level: Beginning Student
Coverage: Module 3

1. An on-line gaming site offers three levels of monthly memberships, which allow you to play a set number of games. Members may play additional games in a month, but must pay additional fees for those extra games. The specifics of the memberships are given below.

Membership Level	Monthly Fee	Games Included	Additional Game Fee
Basic	$ 10.00	50	$ 0.50
Enhanced	$ 15.00	75	$ 0.25
Premium	$ 25.00 (*)	Unlimited	$ 0.00

(*) For the Premium Membership, if you play fewer than 100 games in the month, then there is a 10% discount on monthly fees that month.

Complete a Scheme function `monthly-fees` that consumes a symbol (one of `'Basic`, `'Enhanced`, `'Premium`) and a natural number (the number of games played in the month), and produces the charge for that month. For example, `(monthly-fees 'Enhanced 100) => 21.25`

2. Complete a Scheme function `create-userid` that consumes three lower-case strings (for the first, middle and last names of a person) and a positive integer `index`, and produces a string containing the userid for that person. The userid is made up of the first letter of the first name, the first letter of the second name (if it is not empty), the number `index` (if it is not 1), and the last name, to a maximum of 8 characters. The first and last names will be non-empty, but the

middle name may be empty. For example, `(create-userid "lori" "michelle" “case” 1) => "lmcase"`, and `(create-userid "lionel" "" "snelgrove" 4) => "l4snelgr"`. Do not use the built-in function `string-ref`.

3. One part of the dice game Yahtzee involves throwing five dice repeatedly to determine how many ones, twos, threes, etc. a player can get in different turns. Complete a Scheme function `yahtzee-subtotal` which consumes `ones`, `twos`, `threes`, `fours`, `fives`, `sixes` (all natural numbers between 0 and 5, inclusive) and produces the corresponding total for this part of the game. The total is calculated by multiplying `ones` by 1, `twos` by 2, `threes` by 3, etc. and summing together. The function produces that sum (if it is less than 63), or produces the sum plus a bonus of 35 otherwise. For example, `(yahtzee-subtotal 3 3 3 3 3 3) => 98, (yahtzee-subtotal 1 3 4 4 3 2) => 62`.

4. 6 Nimmt is a card game involving 104 cards (numbered from 1 to 104). One part of the game involves playing a card from your hand, and determining which of 4 piles it must be placed on, if any. If the value of your card is less than the top card in each of the four piles, you cannot place your card. Otherwise, you must place your card on the pile whose top card is smaller than your card, but which is closest in value to your card. For example,

Your card	Top of Pile 1	Top of Pile 2	Top of Pile 3	Top of Pile 4	Place card?
4	78	19	33	34	Cannot place
32	78	19	33	34	Pile 2
53	78	19	33	34	Pile 4
102	78	19	33	34	Pile 1

Complete the Scheme function `place-card` which consumes five parameters: `new-card` (for the card from your hand), and `pile1`, `pile2`, `pile3`, `pile4` (for the top values in each of the 4 piles). All card values are different and between 1 and 104, inclusive. The function produces `'Cannot-place` if `new-card` is less than the other consumed values, or one of `'Pile1`, `'Pile2`, `'Pile3`, `'Pile4`, corresponding to the pile on which `new-card` can be played. For example, `(place-card 32 78 19 33 34) => 'Pile2`.