CSE1729 - Introduction to Programming — February 7, 2017

Laboratory Assignment 3

Objectives

- More work with recursive functions
- Work with the `random` function

Activities

1. The following continued fraction provides an approximation for π:

$$\frac{4}{\pi} = 1 + \cfrac{1^2}{2 + \cfrac{3^2}{2 + \cfrac{5^2}{2 + \cfrac{7^2}{2 + \ddots}}}}$$

Note that the numerator of each continued fraction is the square of the next largest odd number. Also, for the base case, you can use 2 as the denominator in the fraction. For example, if the fraction with $\frac{7^2}{2+\ddots}$ is the final fraction, you can ignore the diagonal dots (there are no recursive calls past this one). Therefore, the last fraction computed would be $\frac{7^2}{2}$. Write a Scheme function to compute this continued fraction. Your function should take one parameter, the number of continued fractions to compute recursively.

Solution:

```
(define (pi-approx-cf k)
  (define (pi-approx-aux i)
    (let* ((ith-odd (+ (* 2  i) 1))
           (numerator (* ith-odd ith-odd)))
      (if (= k i)
          (/ numerator 2)
          (/ numerator (+ 2 (pi-approx-aux (+ i 1)))))))
  (+ 1 (pi-approx-aux 0)))
```

2. Another method of approximating π is achieved through the product of the following nested radicals:

$$\frac{2}{\pi} = \sqrt{\frac{1}{2}} \cdot \sqrt{\frac{1}{2} + \frac{1}{2}\sqrt{\frac{1}{2}}} \cdot \sqrt{\frac{1}{2} + \frac{1}{2}\sqrt{\frac{1}{2} + \frac{1}{2}\sqrt{\frac{1}{2}}}} \cdots$$

Write a Scheme function which computes the product of these nested radicals for a given number of terms, k.

Solution:

```
(define (pi-approx-nrad k)
  (define (pi-approx-aux i)
  (if (= i 0)
      (sqrt (/ 1 2))
      (sqrt (+ (/ 1 2) (* (/ 1 2) (pi-approx-aux (- i 1)))))))
  (if (= k 0)
      (pi-approx-aux 0)
      (* (pi-approx-aux k) (pi-approx-nrad (- k 1)))))
```

3. The game of *Craps* is a dice game in which the player rolls two dice. The player may win the game on the first roll by rolling a 7 or 11. Write a Scheme function that approximates the probability of winning at *Craps* on the first roll by simulating a large number of dice rolls. You will need to use the *random* function to simulate a randomly thrown die. The `random` function takes one integer argument k, and returns a random exact integer in the range 0 to k-1.
For example, to simulate the roll of one die:

```
(random 6)
```

will return an integer in the range 0 to 5. You can add one to obtain roll values in the range 1 to 6. To simulate the rolling of two dice, you will have to use `random` to simulate each die separately and compute the sum of their values.
Your function should take a formal parameter n indicating the number of rolls to simulate and return a decimal representing the percent of rolls won. What win percentage is your simulation approaching as n gets large?

Solution:

```
(define (craps n)
  (define (win-count n wins)
    (if (= n 0) wins
        (let* ((roll1 (+ (random 6) 1))
               (roll2 (+ (random 6) 1))
               (sum (+ roll1 roll2)))
          (if (or (= sum 7) (= sum 11)) (win-count (- n 1) (+ wins 1))
            (win-count (- n 1) wins))
        )
    )
  )
  (exact->inexact (/ (win-count n 0) n))
)
```