

CS-257L

Nonimperative Programming: Scheme!

Instructor:

Joel Castellanos

e-mail: joel@unm.edu

Web: <http://cs.unm.edu/~joel/>

Office: Farris Engineering
Center (FEC) room 321



Homework Due Wednesday, March 5

- Nothing to hand-in.
- Scheme and The Art of Programming
- Read Chapter 4, and do
 - Exercise 4.3
 - Exercise 4.8
 - Exercise 4.14
- Be prepared for a quiz (open notes and open book) on this material.
- This will be like the final exam – you will be given Scheme code and asked to find the return value.

Scheme Do Statement

```
(do ((i 0 (+ i 1))) ((>= i 5))
    (display i)
    (display " "))
)
(newline)
```

Output:

```
0 1 2 3 4
```

roulette (lambda (numBets maxBet)

```
;Check for input errors
(do ((i 0 (+ i 1))) ((>= i numBets))
  ;Place a bet
  (set! myMoney (- myMoney currentBet))
  (cond ((< (random) (/ 18 38))
    ;I win!!!
    (set! myMoney (+ myMoney (* 2 currentBet)))
    (set! currentBet 1))
    (else
     ;I lose
     (set! currentBet (* 2 currentBet))
     (if (> currentBet maxBet)
        (set! currentBet 1))
```

Roulette Results

- Since random numbers control the results of this function, the exact output is not predictable.
- Indeed, for small values of numBets the results are accentually random.
- However, if written correctly, large values of numBets will give consistent results for a given value of maxBet.

Roulette: Check Special Cases

- Change chance of winning to ($/ 38 38$)
(roulette 10 16) → win \$10.00.

- Change chance of winning to ($/ 0 38$)
(roulette 10 1) → lose \$10.00.
(roulette 10 2) → lose \$15.00.
(roulette 10 4) → lose \$22.00.

- Restore chance of winning to ($/ 18 38$)

Set maxBet = 1, then every bet is 1 dollar.

Thus, after 1,000,000 bets, there will be very close to

$1,000,000 * 18/38$ wins and $1,000,000 * 20/38$ losses.

After 1,000,000 bets, it is highly predictable that you will lose between \$50,000 and \$55,000.

Roulette: Complexity for higher values of MaxBet

- When maxBet is 1, then every bet is an independent event.
- However, if maxBet is 512, then a single event may be composed of as many as 10 bets.
- Thus, as maxBet gets larger, a given number of bets will consist of less events and, therefore, have less predictable results.
- Yet, for large enough values of numBets, the results will be highly predictable.

Roulette: Expected Results

maxBet	Expected Results with numBets = 1,000,000
1	Loose between \$50,000 and \$55,000.
2	Loose between \$50,000 and \$100,000.
5,000	Loose between \$250,000 and \$1,500,000.
50,000	Loose between \$200,000 and \$2,000,000.
5,000,000	Win between \$450,000 and \$500,000 (with 1 in 29 chance of loosing 4 million).
50,000,000	Win between \$450,000 and \$500,000 (with 1 in 273 chance of loosing 34 million).

- In the last two cases, the result is winning almost half a million dollars.
- This is dependant on there NEVER being a breaking loosing streak.
- A single breaking loosing streak at this magnitude, cannot be recovered in 1 million bets even if every other bet is a win on the first try.

Roulette: Probabilities

- The rate at which the bet increases is exponential: (2^n) .
- The Probability of a catastrophic loosing streak decreases exponentially: $(18/38)^n$.

n	$2^{(n-1)}$	$(18/38)^{(n-1)}$
1	1	4.7 E-1
2	2	2.2 E-1
3	4	1.1 E-1
4	8	5.0 E-2
5	16	2.4 E-2

n	$2^{(n-1)}$	$(18/38)^{(n-1)}$
15	16,384	1.4 E-5
20	524,288	3.2 E-7
23	4,194,304	3.4 E-8
26	33,554,432	3.7 E-9
29	268,435,456	3.9 E-10

Roulette: Probabilities

n	MaxBet = $2^{(n-1)}$	Prob. of loosing streak $(18/38)^{(n-1)}$
26	33,554,432	3.7 E-9
29	268,435,456	3.9 E-10

- Thus, with 1 million bets and a \$34 million bet limit, there is a:
 - $1,000,000 * 3.7E-9 (=1/273)$ chance of loosing 34 million and a
 - $(272/273)$ chance of winning about $1,000,000 * (18/38) = \$470,000$.
- With a max bet of 269 million, the chance of loosing after 1 million bets is $1,000,000 * 3.9E-10 (=1/2500)$.
- As maxBet gets larger, the chance of loosing decreases a bit faster than the cost of loosing.

Recursion Vocabulary

- Flat Recursion

Recursion being done over the top-level items in lists:

`(flatCount '(a (x y z) b))` → 3

- Deep Recursion

- When recursion is over all the atomic items of a list:

`(deepCount '(a (x y z) b))` → 5

More Vocabulary

- We say that the sublist (b c) is *nested* in list (a (b c)).
- If an item is not enclosed in parentheses, that item has *nesting level 0*.
- The item c in (a (b (c d))) has nesting level 3.