CS3:
Introduction to Symbolic Programming

Lecture 11:
Tree Recursion, beginning lists, and Midterm 2

Spring 2007

Nate Titterton
nate@berkeley.edu

Schedule

11	April 2-6	Lecture: Midterm review, tree recursion Lab: Lists, tree-recursion ~~Miniproject #3 due Tuesday~~
12	April 9-13	Lecture (5-7 pm): *Midterm #2* -- 145 Dwinelle Lab: Advanced list processing
13	April 16-20	Lecture: CS3 Projects, Lists Lab: Begin work on CS3 Big Project Non-functional programming ~~Reading: Simply Scheme, chapter 20~~
14	April 23-27	Lecture: Advanced lists, project review Lab: Work on projects
15	Apr 30-May 4	Lecture: CS at Berkeley (guest lecture) Lab: Finish projects (due end of week)
16	May 7	Lecture: Exam review *no more labs!*

Any questions about the miniproject?

Midterm 2

Announcements

- **Midterm 2 is coming…**
  - Next week, 80 minutes (5:10-6:30).
  - Room **145 Dwinelle**
  - Open book, open notes, etc.
  - Check for practice exams and solution on the course portal and in the reader.

- **Midterm 2 review session**
  - Sunday, Apr. 8, 4-6pm in 430 Soda.

What does midterm #2 cover?

- Advanced recursion (accumulating, multiple arguments, etc.).
- Tree-recursion (from this week)
- All of higher order functions
- Those "big" homeworks (bowling, compress, and occurs-in)
- Elections and number-name miniproject
- Reading and programs:
  - Change making, Roman numerals
  - Difference between dates #3 (HOF),
  - tic-tac-toe
- SS chapters 14, 15, 7, 8, 9, 10
- Everything before the first Midterm (although, this won't be the focus of a question)

Tree recursion

(coming this week)

What will happen?

- What will `countem` return for n=1, 2, …?

```
(define (countem n)
   (if (= n 0)
       '()
       (se (countem (- n 1))
           n
           (countem (- n 1)))))
```

Tree recursion

A recursive technique in which more than one recursive call is made within a recursive case.

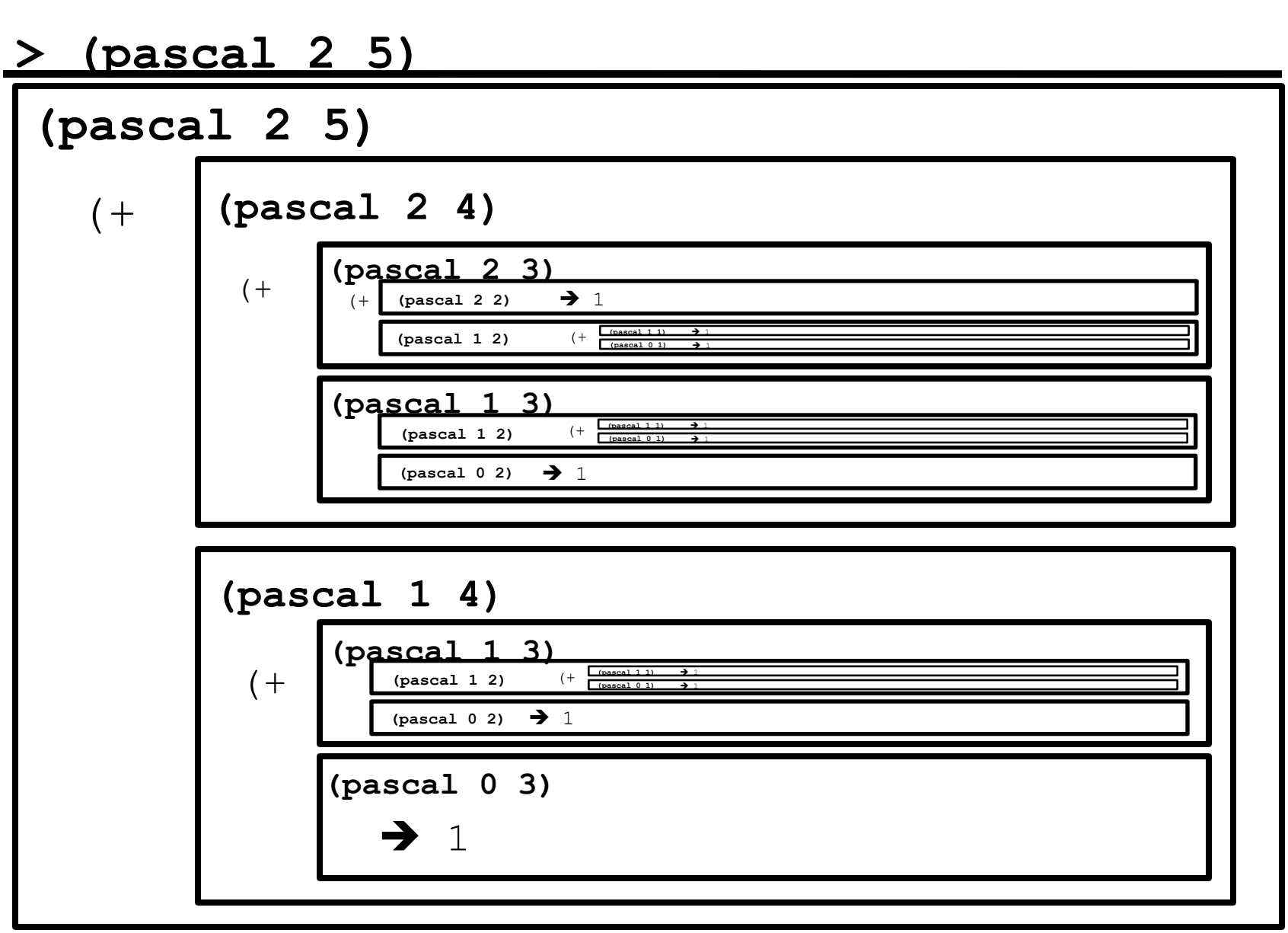
Pascal's triangle

	columns (C)							
rows (R)		0	1	2	3	4	5	...
	0	1						...
	1	1	1					...
	2	1	2	1				...
	3	1	3	3	1			...
	4	1	4	6	4	1		...
	5	1	5	10	10	5	1	...
	...	...	...	...	...	...	...	...

Pascal's Triangle

- How many ways can you choose C things from R choices?
- Coefficients of the (x+y)^R: look in row R
- etc.

```
(define (pascal C R)
  (cond
    ((= C 0) 1)    ;base case
    ((= C R) 1)    ;base case
    (else          ;tree recurse
     (+ (pascal     C     (- R 1))
        (pascal (- C 1) (- R 1))
     )))
```

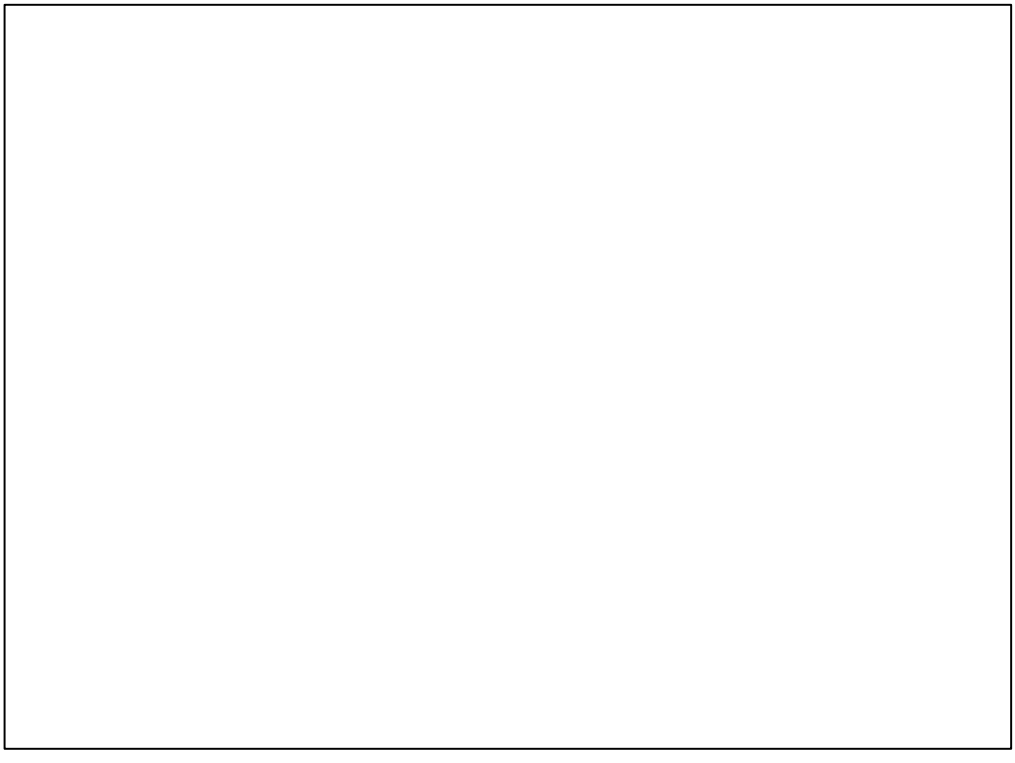

> (pascal 2 5)
(pascal 2 5)
(+
(pascal 2 4)
(+
(pascal 2 3)
(+
(pascal 2 2)
➔ 1
(pascal 1 2)
(+
(pascal 1 1)
➔ 1
(pascal 0 1)
➔ 1
(pascal 1 3)
(pascal 1 2)
(+
(pascal 1 1)
➔ 1
(pascal 0 1)
➔ 1
(pascal 0 2)
➔ 1
(pascal 1 4)
(+
(pascal 1 3)
(pascal 1 2)
(+
(pascal 1 1)
➔ 1
(pascal 0 1)
➔ 1
(pascal 0 2)
➔ 1
(pascal 0 3)
➔ 1

Problems

binary

- **Write `binary`, a procedure to generate the possible binary numbers given `n` bits.**

```
(binary 1)→(0 1)
(binary 2)→(00 01 10 11)
(binary 3)→(000 001 010 011 100 101 110 111)
```



Schedule

11	April 2-6	Lecture: Midterm review, tree recursion Lab: Lists, tree-recursion Miniproject #3 due Tuesday
12	April 9-13	Lecture (5-7 pm): *Midterm #2* -- 145 Dwinelle Lab: Advanced list processing
13	April 16-20	Lecture: CS3 Projects, Lists Lab: Begin work on CS3 Big Project Non-functional programming Reading: Simply Scheme, chapter 20
14	April 23-27	Lecture: Advanced lists, project review Lab: Work on projects
15	Apr 30-May 4	Lecture: CS at Berkeley (guest lecture) Lab: Finish projects (due end of week)
16	May 7	Lecture: Exam review *no more labs!*

Any questions about the miniproject?

Click to add text

Midterm 2

Click to add text

Announcements

- **Midterm 2 is coming…**
  - Next week, 80 minutes (5:10-6:30).
  - Room **145 Dwinelle**
  - Open book, open notes, etc.
  - Check for practice exams and solution on the course portal and in the reader.

- **Midterm 2 review session**
  - Sunday, Apr. 8, 4-6pm in 430 Soda.

What does midterm #2 cover?

- **Advanced recursion (accumulating, multiple arguments, etc.).**
- **Tree-recursion (from this week)**
- **All of higher order functions**
- **Those "big" homeworks (bowling, compress, and occurs-in)**
- **Elections and number-name miniproject**
- **Reading and programs:**
  - **Change making, Roman numerals**
  - **Difference between dates #3 (HOF),**
  - **tic-tac-toe**
- **SS chapters 14, 15, 7, 8, 9, 10**
- **Everything before the first Midterm (although, this won't be the focus of a question)**

Tree recursion

(coming this week)

What will happen?

- **What will `countem` return for n=1, 2, …?**

```
(define (countem n)
   (if (= n 0)
       '()
       (se (countem (- n 1))
           n
           (countem (- n 1)))))
```

STk> (countem 1)

(1)

STk> (countem 2)

(1 2 1)

STk> (countem 3)

(1 2 1 3 1 2 1)

STk> (countem 4)

(1 2 1 3 1 2 1 4 1 2 1 3 1 2 1)

Tree recursion

A recursive technique in which more than one recursive call is made within a recursive case.

Pascal's triangle

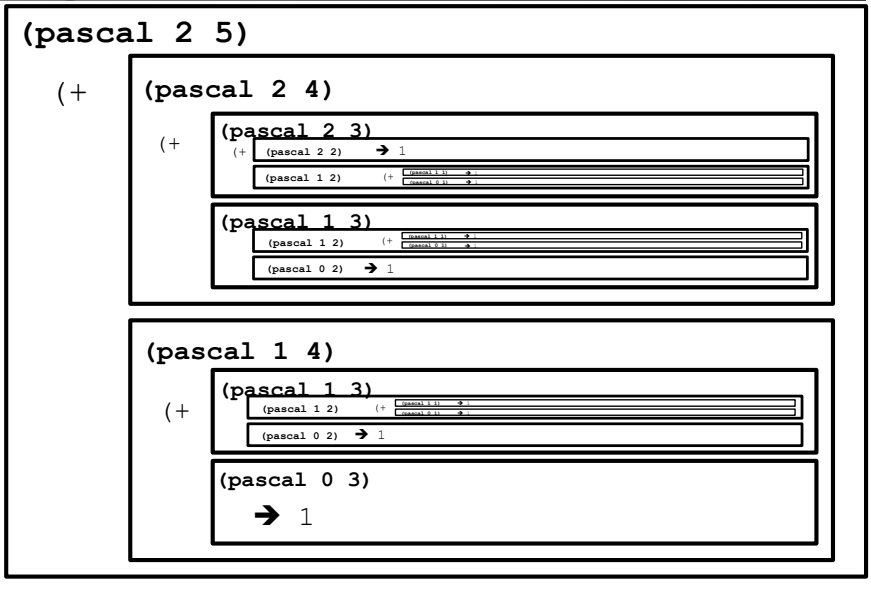
	columns (C)							
rows (R)		0	1	2	3	4	5	...
	0	1						...
	1	1	1					...
	2	1	2	1				...
	3	1	3	3	1			...
	4	1	4	6	4	1		...
	5	1	5	10	10	5	1	...
	...	...	...	...	...	...	...	...

Pascal's Triangle

- How many ways can you choose C things from R choices?
- Coefficients of the (x+y)^R: look in row R
- etc.

```
(define (pascal C R)
  (cond
    ((= C 0) 1)    ;base case
    ((= C R) 1)    ;base case
    (else          ;tree recurse
     (+ (pascal    C    (- R 1))
        (pascal (- C 1) (- R 1))
     )))
```

> (pascal 2 5)


(pascal 2 5)
(+
(pascal 2 4)
(+
(pascal 2 3)
(+ (pascal 2 2) → 1
(pascal 1 2) (+
(pascal 1 3)
(pascal 1 2) (+
(pascal 0 2) → 1
(pascal 1 4)
(+
(pascal 1 3)
(pascal 1 2) (+
(pascal 0 2) → 1
(pascal 0 3)
→ 1

Problems

Click to add text

binary

- **Write `binary`, a procedure to generate the possible binary numbers given `n` bits.**

```
(binary 1)→(0 1)
(binary 2)→(00 01 10 11)
(binary 3)→(000 001 010 011 100 101 110 111)
```

```
(define (binary n)
  (if (= n 1)
      '(0 1)
      (se (prepend-every 0 (binary (- n 1)))
          (prepend-every 1 (binary (- n 1))))))

(define (prepend-every what sent)
        (if (empty? sent) '()
          (se (word what (first sent))
              (prepend-every what (bf sent)))))
```