#### Homework

- Next homework out this afternoon
- Full-length assignment

#### Lab lecture

- Focus on programs with multiple complex arguments
- This stuff is important
  - $\rightarrow$  many real programs fit this mold

COMP 210, Spring 2002

## Programs with Multiple Complex Arguments

So far, three cases

- Two arguments, one is not inspected
  - $\rightarrow$  Use template for the inspected argument
- Two arguments, with simplifying property
  - $\rightarrow$  Lists of same length
  - $\rightarrow\,$  Trees of identical shape
  - $\rightarrow\,$  Use one argument to control the flow of the program
- Two arguments, no simplifying assumptions
  - $\rightarrow$  Build a table of the cases
  - $\rightarrow\,$  Develop tests for each case
  - $\rightarrow\,$  Use a cond with a clause for each case
  - $\rightarrow$  Lots of opportunities to recur





Example: append

Example: make-points

1



Another example

;; merge : list-of-numbers list-of-numbers -> list-of-numbers ;; Purpose: consumes two lists of numbers, assumed to be in ;; ascending order by value, and produces a single list of ;; numbers that contains all the elements of the input lists ;; (including duplicates) in ascending order by value (define (merge a-lon1 a-lon2) ...)

- Merge must look inside both lists
- The lists can have different length

 $\rightarrow$  (merge empty (cons 1 empty)) should be (cons 1 empty)

COMP 210, Spring 2002

Programs with Multiple Complex Arguments



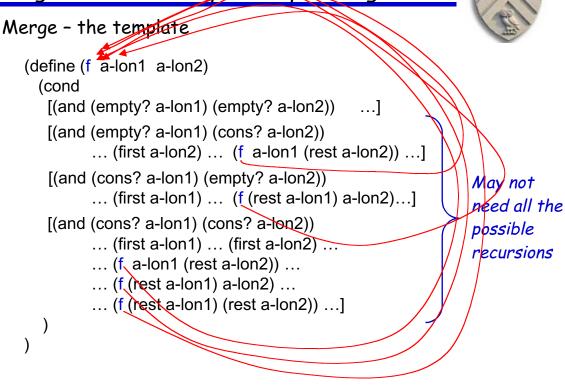
3

#### Merge

Questions for list x list

	(empty? a-lon2)	(cons? a-lon2)
(empty? a-lon1)	(and (empty? a-lon1) (empty? a-lon2))	(and (empty? a-lon1) (cons? a-lon2))
(cons? a-lon1)	(and (cons? a-lon1) (empty? a-lon2))	(and (cons? a-lon1) (cons? a-lon2))

The template must include (and handle) all these cases



```
COMP 210, Spring 2002
```

#### Programs with Multiple Complex Arguments



5

## Merge - the program

```
(define (merge a-lon1 a-lon2)
 (cond
   [(and (empty? a-lon1) (empty? a-lon2))
                                               empty]
   [(and (empty? a-lon1) (cons? a-lon2))
                                               a-lon2]
   [(and (cons? a-lon1) (empty? a-lon2))
                                               a-lon1]
   [(and (cons? a-lon1) (cons? a-lon2))
    (cond
        [(< (first a-lon1) (first a-lon2)
         (cons (first a-lon1) (merge (rest a-lon1) a-lon2))]
         [else
          (cons (first a-lon2) (merge a-lon1 (rest a-lon2)))]
     )
   ]
)
```

What good is merge?

- Forms the core of a general algorithm for sorting •
- To sort a list •
  - $\rightarrow$  Break list into lists of length one
  - $\rightarrow$  Merge adjacent lists, merge results, ...

The result is method of choice for sorting sets of data that are too large to fit in memory

COMP 210, Spring 2002

Programs with Multiple Complex Arguments					
Sorting with me	rge		(II)		
(list c1 c2 c3 c4	c5 c6 c7 c8) $\Rightarrow$				
↓ merge	(list c3) (list c4) ↓ merge (list ck cl)	↓ merge	↓ merge		
↓ merge (list ci cj ck cl)		↓ merge (list cm cn co cp)			
	Ų	merge			
	(list ci cj ck cl d	cm cn co cp)			

Question becomes, can we generate singleton lists from a list?

- We do not yet have the tools to do this
- Next section of 210 examines a paradigm that can do this



(or two)



7



Methodology

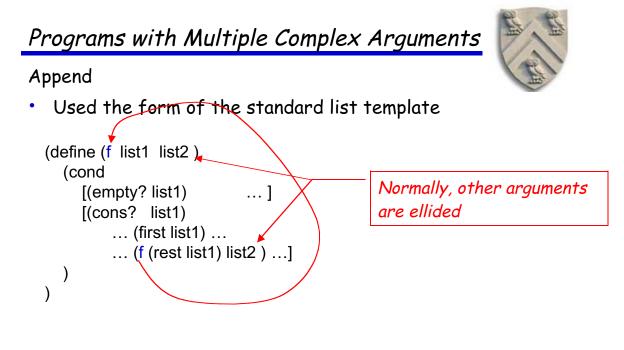
# Wrapping up this set of ideas

COMP 210, Spring 2002

## Programs with Multiple Complex Arguments

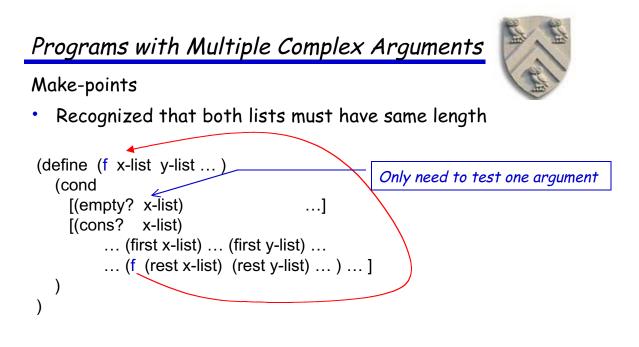
Philosophy

- In general, there is only one template for a pair of arguments
- For list x list, it's the full template we developed for merge
- We may, however, simplify that template
  - $\rightarrow$  Problem-specific knowledge, as in <code>append</code> or <code>make-points</code>
  - $\rightarrow$  These simplified templates speed up development
  - $\rightarrow$  These simplified templates may lead to cleaner programs



• Key was to recognize that list2 is uninspected

COMP 210, Spring 2002



• Simplifies the cond structure

11



#### Merge



### Programs with Multiple Complex Arguments

Philosophy

- In general, there is only one template for a pair of arguments
- For list x list, it's the full template we developed for **merge**
- We may, however, simplify that template
  - $\rightarrow$  Problem-specific knowledge, as in **append** or **make-points**
  - $\rightarrow$  These simplified templates speed up development
  - $\rightarrow$  These simplified templates may lead to cleaner programs

#### But,

They are still special cases of the general template