



Trees



Today's Quiz

- Max function
 - Mentioned (at least) in 2.1.1
- Conventions for types & Tree problem
 - Covered in book
 - We will spend time on these points today

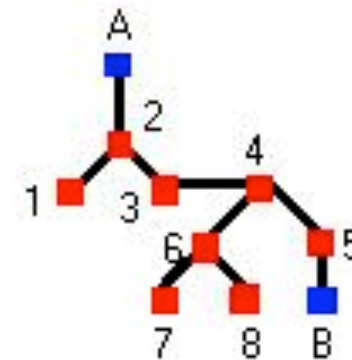
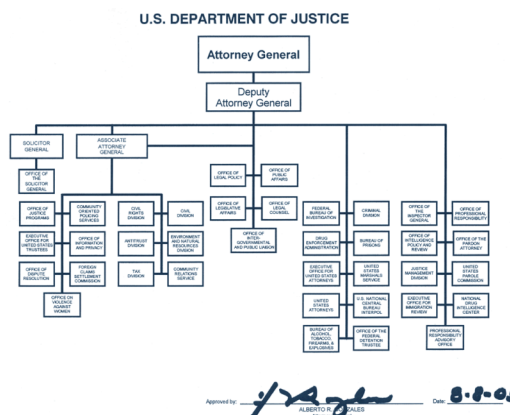
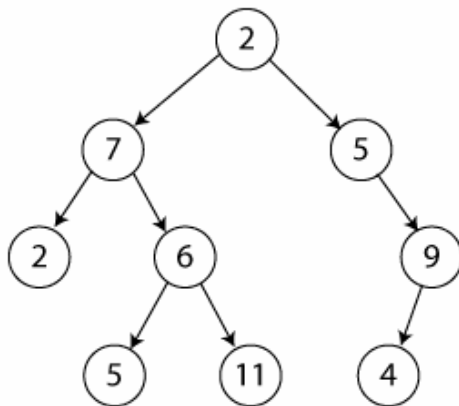


Today's Goals

- Trees
 - Significantly more expressive type
 - “Lists with many tails”
- Conventions about type definitions
- Examples:
 - Family tree
 - Binary search tree
 - Web pages

Trees and their applications

- Labeled trees
 - Organizational charts
 - Decision trees
 - Search trees
- and many more!





From Lists to Trees

```
; A list-of-symbols (los for short) is  
;   empty, or  
;   (cons s l)  
; where s is a symbol and l is a los
```

```
; A person is  
;   false // Represents “unknown”  
;   (make-person n m d) // Note: Two “rests”  
; where n is a symbol, and m and d are persons  
(define-struct person (name mom dad))
```



Example Tree

```
(make-person 'Bob
  (make-person 'Jane false
    (make-person 'Tom
      (make-person 'Cat false false) false))
  (make-person 'Rob false
    (make-person 'Sue false
      (make-person 'Ray false
        (make-person 'Johny false false))))))
```



Template for a Tree (1/3)

- We first test for which variety

```
; f : person -> ...  
; (define (f x)  
;   (cond  
;     [(boolean? x) ...]  
;     [else ...
```



Template for a Tree (2/3)

- We make sure that each field is used

```
; f : person -> ...
; (define (f x)
;   (cond
;     [(boolean? x) ...]
;     [else ... (person-name x)
;               (person-mom x) ...
;               (person-dad x) ...])
```



Template for a Tree (3/3)

- Recursion in type \rightarrow recursion in template

```
; f : person  $\rightarrow$  ...  
; (define (f x)  
;   (cond  
;     [(boolean? x) ...]  
;     [else ... (person-name x)  
;               ... (f (person-mom x))...  
;               ... (f (person-dad x))...])
```



Tree Depth (in class ex.)

- Consider the following problem
 - “Given a person tree, compute the maximum number of generations for which we know something about this person.”
- Contract (or “type”) is
 - `person -> natural`
- Examples (from above)
- Template?



Tree Depth

```
f : person -> natural
(define (f x)
  (cond
    [(boolean? x) 0]
    [else (+ 1
             (max (f (person-mom x))
                    (f (person-dad x))))])
```

Examples were really helpful for filling in the code!

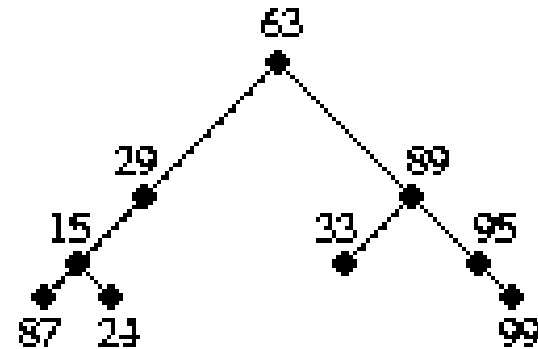
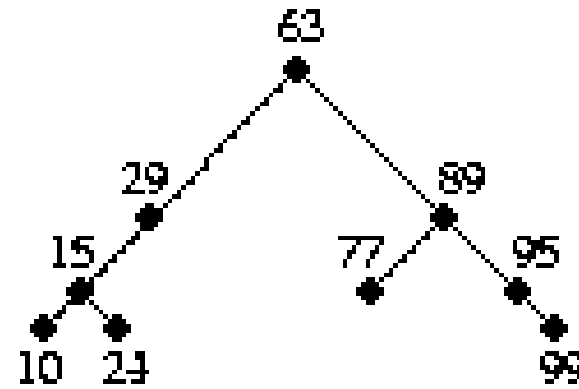
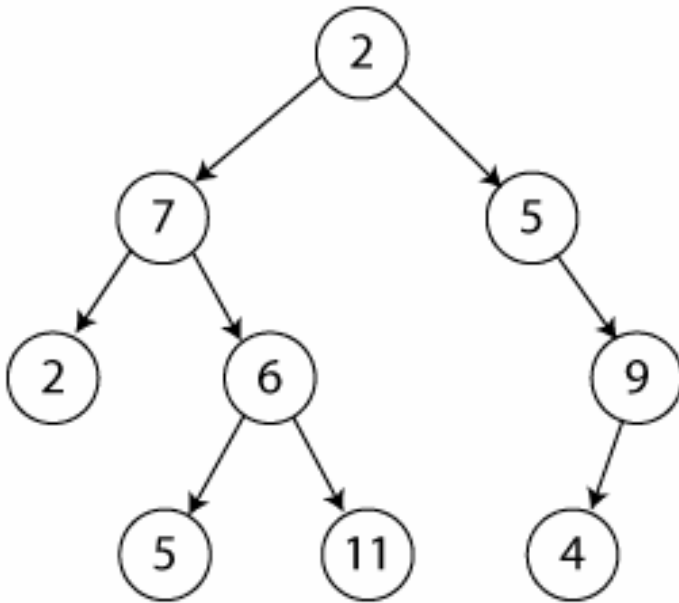


Conventions for type definitions

- Our type definitions have a very specific form. Does this definition fit in?

```
    ; A child node is (make-child f m na da ec)  
    ; where f and m are either  
    ;     empty or  
    ;     child nodes
```
- Our convention:
 - The new type is a variety of values (forms)
 - The type of each field in each struct is a name

Binary Search Trees



Which tree is different?



Binary Search Trees

```
; A binary-tree (BT) is either
;   false, or
;   (make-node n l r)
; where n is a number, l and r are BTs.
; Invariants:
;   1. Numbers in l are less than or equal to n
;   2. Numbers in r are greater than n
(define-struct node (num left right))
```



Web Pages

```
; A web-page (WP) is either  
;   empty, or  
;   (cons s wp), or  
;   (cons wp1 wp2),  
; where s is a symbol,  
;   and wp, wp1, and wp2 are WP
```



For Next Class

- Homework due Monday
- Midterm:
 - Wednesday, in class
 - Chapters 1-13
- No quizzes until midterm