"Three-Eyed Aliens Stole My Lecture Notes," claims CS Prof.

(Claremont AP): A distraught CS professor at Harvey Mudd College claims that aliens broke into his office last night and stole the notes that he planned to use in CS 42. "These aliens made quite a Racket! When I came out to evaluate, they cond me into lending them my keys, went into my office, made a list of the contents, and took the first item on that list in an effort to null my lecture," said the professor. "Our first task is to map the crime," said a detective on the scene. "These aliens must recognize the cons of stealing CS 42 notes. They should know that we will not rest or foldr our investigation until we've appended every last culprit." Evidence at the crime scene included a number of items that could not be defined as well as several mismatched parentheses. "We're filtering through it," said investigators. Why are you reading this drivel?! You should be paying attention to lecture!

Trees

September 8, 2011

CS 42: Principles and Practice of Computer Science

Thinking Functionally

After the entire sequence:

```
(define x '(1 2 3))
(define y (cons 14 x))
(define z (append '(6 7) x))
(define w (reverse x))
(define n (foldr + 10 x))
(map (lambda (q) (+ q 1)) x)
```

What are the values of `x`, `y`, `z`, `w`, and `n`?

`lotto`: PLANNING

Given tickets `'( (name n1 n2 n3 n4 n5) (name n1 n2 n3 n4 n5) ... )` and a list of winning numbers, `W` return the name of the winner (most matches with `W`) and the number of matches made.

```
> (lotto '( (Amy 2 3 41 42 50) (Bea 3 40 50 51 52)) '( 1 3 41 51 52) )
'(Bea 3)
```

`lotto`: PLANNING

Given tickets `'( (name n1 n2 n3 n4 n5) (name n1 n2 n3 n4 n5) ... )` and a list of winning numbers, `W` return the name of the winner (most matches with `W`) and the number of matches made.

```
> (lotto '( (Amy 2 3 41 42 50) (Bea 3 40 50 51 52)) '( 1 3 41 51 52) )
'(Bea 3)
```

Suggested Strategy:

- ✓ Write a function `(matches L1 L2)` that returns the number of elements in `L1` also in `L2`.

`lotto`: PLANNING

Given tickets `'( (name n1 n2 n3 n4 n5) (name n1 n2 n3 n4 n5) ... )` and a list of winning numbers, `W` return the name of the winner (most matches with `W`) and the number of matches made.

```
> (lotto '( (Amy 2 3 41 42 50) (Bea 3 40 50 51 52)) '( 1 3 41 51 52) )
'(Bea 3)
```

Suggested Strategy:

- ✓ Write a function `(matches L1 L2)` that returns the number of elements in `L1` also in `L2`.
  - ▸ Create a `1` or `0` for each element in `L1`, depending on matching.

`lotto`: PLANNING

Given tickets `'( (name n1 n2 n3 n4 n5) (name n1 n2 n3 n4 n5) ... )` and a list of winning numbers, `W` return the name of the winner (most matches with `W`) and the number of matches made.

```
> (lotto '( (Amy 2 3 41 42 50) (Bea 3 40 50 51 52)) '( 1 3 41 51 52) )
'(Bea 3)
```

Suggested Strategy:

- ✓ Write a function `(matches L1 L2)` that returns the number of elements in `L1` also in `L2`.
  - Create a `1` or `0` for each element in `L1`, depending on matching.
  - Sum these numbers.

`lotto`: PLANNING

Given tickets `'( (name n1 n2 n3 n4 n5) (name n1 n2 n3 n4 n5) ... )` and a list of winning numbers, `W` return the name of the winner (most matches with `W`) and the number of matches made.

```
> (lotto '( (Amy 2 3 41 42 50) (Bea 3 40 50 51 52)) '( 1 3 41 51 52) )
'(Bea 3)
```

Suggested Strategy:

- ✓ Write a function `(matches L1 L2)` that returns the number of elements in `L1` also in `L2`.
  - Create a `1` or `0` for each element in `L1`, depending on matching.
  - Sum these numbers.
  - (Alternatively, you could use `foldr`, or `filter` and `length`.)

`lotto`: PLANNING

Given tickets `'( (name n1 n2 n3 n4 n5) (name n1 n2 n3 n4 n5) ... )` and a list of winning numbers, `W` return the name of the winner (most matches with `W`) and the number of matches made.

```
> (lotto '( (Amy 2 3 41 42 50) (Bea 3 40 50 51 52)) '( 1 3 41 51 52) )
'(Bea 3)
```

Suggested Strategy:

- ✓ Write a function `(matches L1 L2)` that returns the number of elements in `L1` also in `L2`.
  - Create a `1` or `0` for each element in `L1`, depending on matching.
  - Sum these numbers.
  - (Alternatively, you could use `foldr`, or `filter` and `length`.)
- ✓ Write a function `score-em` that computes `(name score)` pairs for everyone.

`lotto`: PLANNING

Given tickets `'( (name n1 n2 n3 n4 n5) (name n1 n2 n3 n4 n5) ... )` and a list of winning numbers, `W` return the name of the winner (most matches with `W`) and the number of matches made.

```
> (lotto '( (Amy 2 3 41 42 50) (Bea 3 40 50 51 52)) '( 1 3 41 51 52) )
'(Bea 3)
```

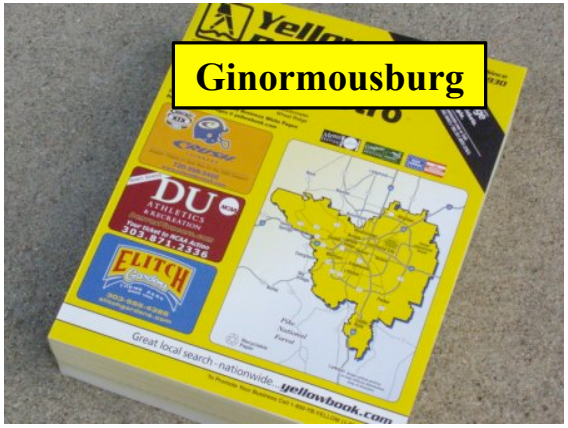
Suggested Strategy:

- ✓ Write a function `(matches L1 L2)` that returns the number of elements in `L1` also in `L2`.
  - Create a `1` or `0` for each element in `L1`, depending on matching.
  - Sum these numbers.
  - (Alternatively, you could use `foldr`, or `filter` and `length`.)
- ✓ Write a function `score-em` that computes `(name score)` pairs for everyone.
- ✓ Write a function `lotto` that picks out the best pair.

lotto: CODE

```
(define (lotto TL W)
   (foldl (lambda (s1 s2)
            (if (> (second s1) (second s2)) s1 s2))
          '(Chris 0)
          (score-em TL W)))

(define (score-em TL W)
   (map (lambda (L) (score L W)) TL))

(define (score L W)
   (list (first L) (matches (rest L) W)))

(define (matches L1 L2)
   (foldr + 0 (map (lambda (n) (markit n L2)) L1)))

(define (markit elem L)
   (if (member elem L) 1 0))
```

Searching Through a Phone Book



2^{30} entries

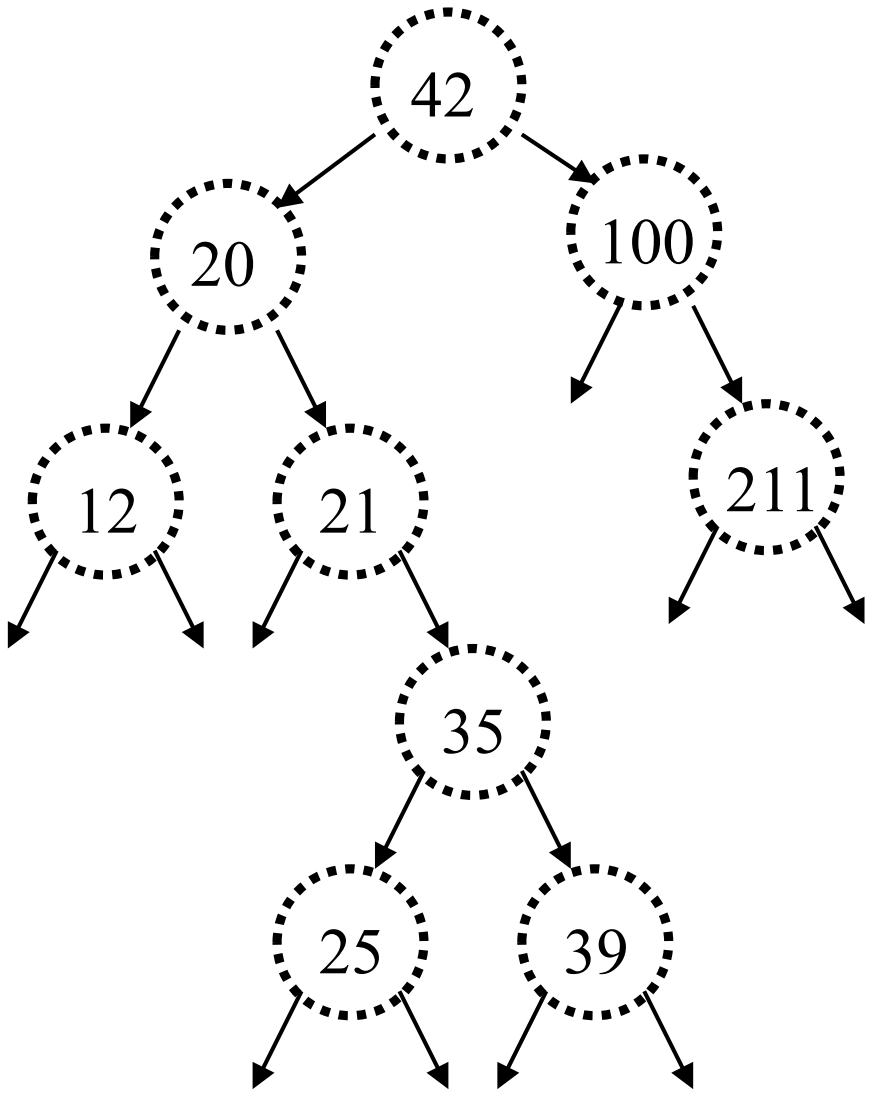
```
((“AAA Aardvark Training” “909-555-NICE”)
 (“AAA Alligators” “909-555-BITE”)
  ...
 (“Zyzzyva Exterminators” 909-555-GONE”))
```

A More Efficient Representation

- ✓ Nodes have up to two children
- ✓ Left subtrees have smaller keys; right subtrees have bigger keys.

Search on the key,
Return the value!

Binary Search Trees

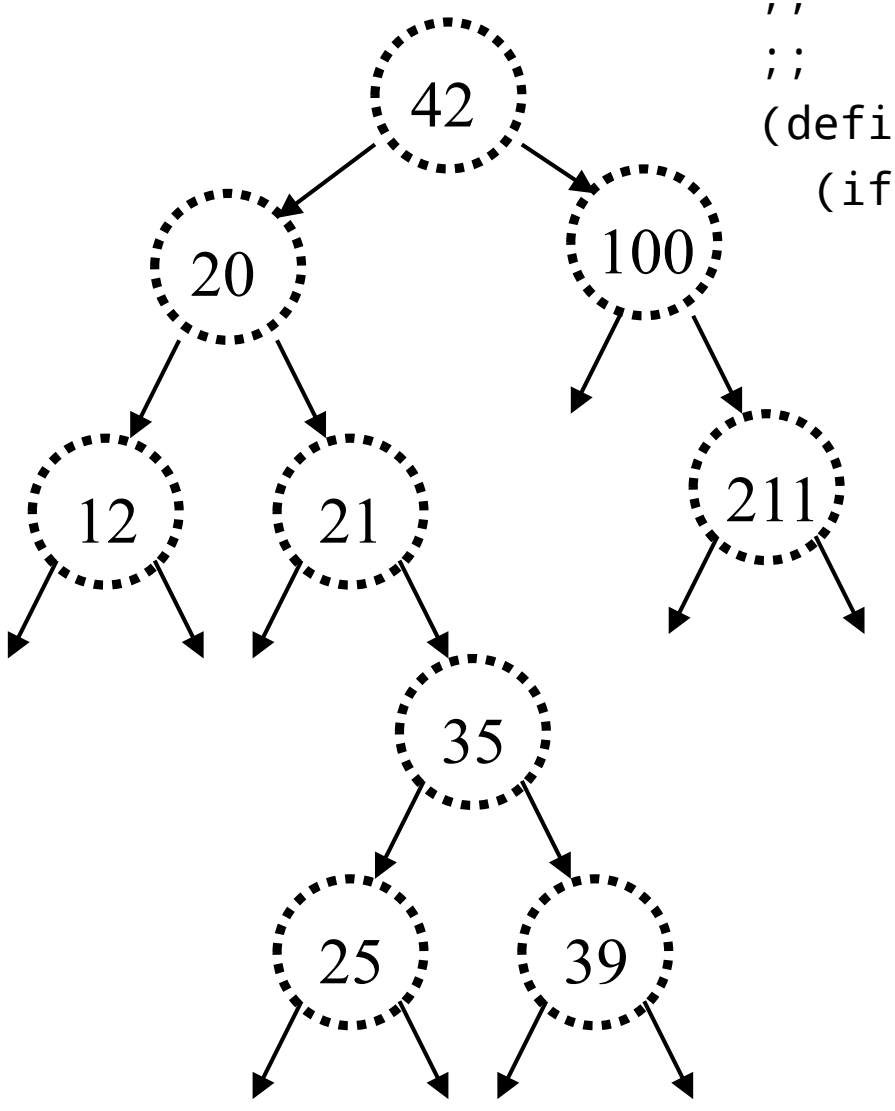


Identifying Features:

- ✓ Every node has two subtrees
- ✓ Each node has a "key"
- ✓ The root key is always greater than all nodes in a left subtree
- ✓ The root key is always less than all nodes in a right subtree

But Racket only has lists...?

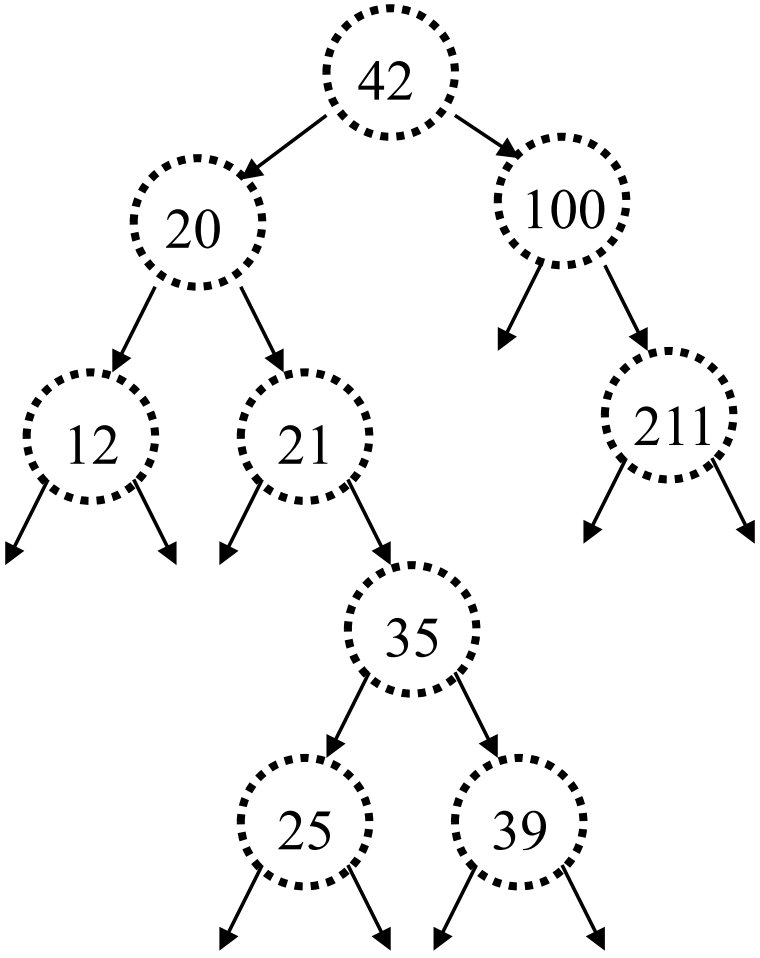
Searching in BSTs




```
;; (find item BST) ===>
;;    #t if item in BST,
;;    #f otherwise
(define (find item BST)
  (if (null? BST)

      (let ( [ root (first BST) ]
             [ LEFT (second BST) ]
             [ RIGHT (third BST) ])
           (
```

"I wonder about Trees…" — Robert Frost

"We wonder about Robert Frost…" — Trees



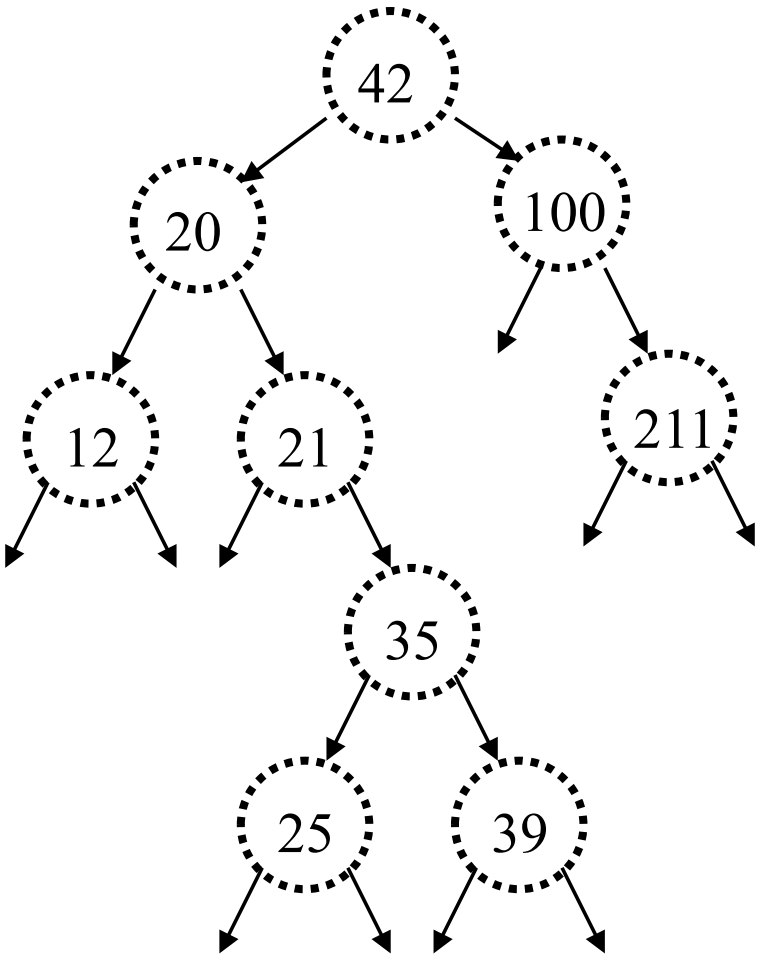
(12 20 21 25 35 39 42 10 211)

Turning BSTs into Ordered Lists

```
(define (traverse BST)
  (if (null? BST)

      (let ( [ root (first BST) ]
             [ LEFT (second BST) ]
             [ RIGHT (third BST) ])
           (
```

Finding the Minimum/Maximum in a BST

Turning BSTs into Ordered Lists




```
;; Assume input is non-empty
(define (findmin BST)
```

(12 20 21 25 35 39 42 10 211)

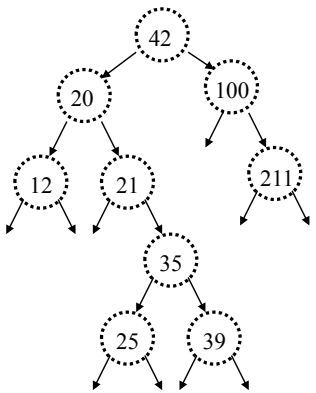
Name:

"Quiz:" Two functions for BSTs

```
;; Given a BST, count the number of nodes
(define (size BST)
```

```
;; Given a BST and a new item not there,
;;   create a new BST containing everything
;;   in the given BST plus the new item.
(define (insert item BST)
```

```
  (let ( [ root (first BST) ]
         [ LEFT (second BST) ]
         [ RIGHT (third BST) ])
    (
```



Trust in recursion…

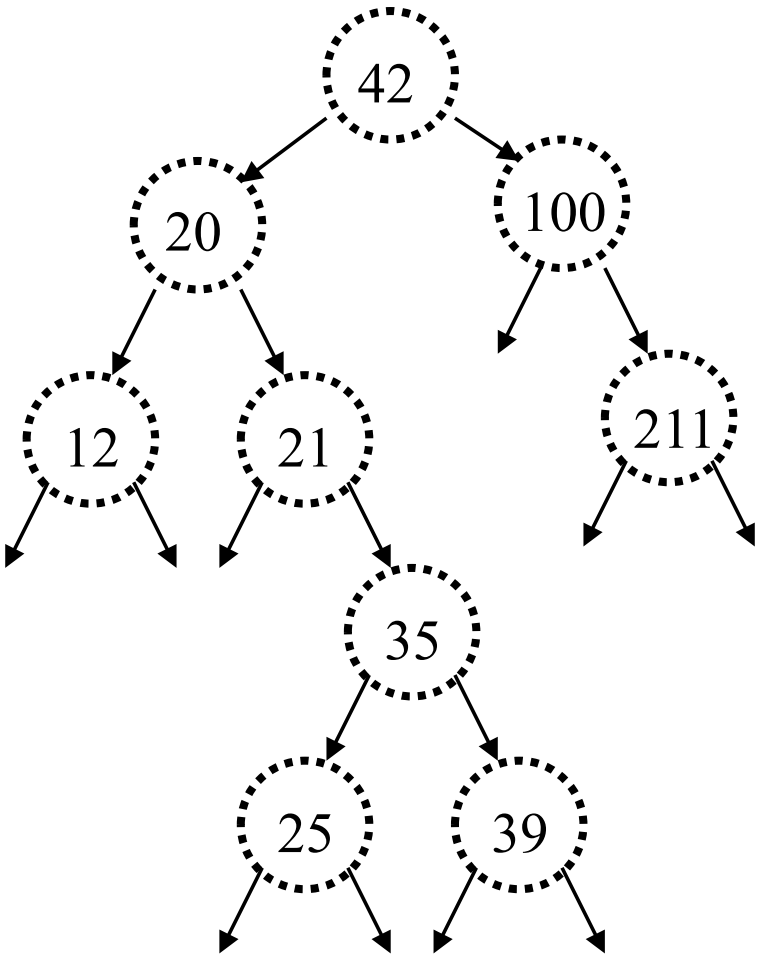
size

```
(define (size BST)
   (if (null? BST)
       0
       (let ( [ LEFT  (second BST) ]
              [ RIGHT (third BST)  ] )
            (+ 1 (size LEFT) (size RIGHT)))))
```

insert

```
(define (insert item BST)
   (if (null? BST)
       (list item '() '())
       (let ( [ root  (first BST)  ]
              [ LEFT  (second BST) ]
              [ RIGHT (third BST)  ] )
            (if (< item root)
                (list root (insert item LEFT) RIGHT)
                (list root LEFT (insert item RIGHT))))
```

Deleting!



To delete an item N:

- ✓ If the tree is empty?
- ✓ If the root is bigger than N?
- ✓ If the root is less than N?
- ✓ If the root is N?
  - If $N = 25$?
  - If $N = 21$?
  - IF $N = 42$?