Sunday, February 17, 2019 10:06 AM

Overview:

Exam this Friday 5-7 PM
↳ all info posted on Piazza
No lecture Friday

Quiz 5 due tonight

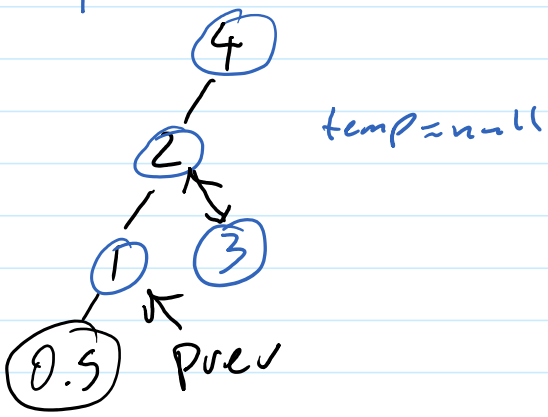
Today:

- finish BST insert()
- recursive functions
- tree traversals

From Last Time:

Insert()

Given an existing tree, insert a node w/ key = 3



1) Create new node (4) and assign key (3)

2) Two pointers for traversal
   temp = root
   prev = null

3) Drill down to find next available empty spot

   while (temp != NULL)

```
prev = temp

// check which way to
// traverse
if (n->key < temp->key)        3 ? 4 True
  ->temp = temp->leftChild <-  3 ? 2 False

else // >=
  ->temp = temp->rightChild <-
```

// found where to put new node
// established parent for new node

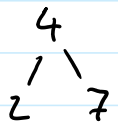
4) Add new node to correct place

```
if prev == NULL
  ↳ means tree was empty
  ↳ while loop ran zero times
  ↳ make new node root

    done

else if (n->key < prev->key)      3 ? 2 F
  ↳ if new key is smaller than
    parent, make new node true
    left child
    prev->leftChild = n;

    n->parent = prev;

else                              3 ? 2 true
  ↳ new key is ≥ parent key
    prev->rightChild = n;

    n->parent = prev;
```

Traversals:

How do we decide about the order?

3 conventions



Pre-order: root, left, right
4, 2, 7

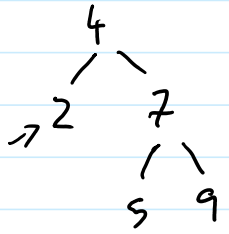
In-order: left, root, right
2, 4, 7

Post-order: left, right, root
2, 7, 4

Pre-order: 4, 2, 7, 5, 9

In-order: 2, 4, 5, 7, 9

Post-order: 2, 5, 9, 7, 4



How to implement?

recursion

C++ allows a function to call another instance of itself.

↳ recursion

```
int foo(int x){
    return foo(x);
}
```

← infinite loop = stack overflow

For any recursive algorithm, a base case needs to be defined.

Once base case is reached, no more recursive calls are made.

↳ the algo can end execution

E.G.

n! n-factorial recursive function

$n! = 1 \times 2 \times 3 \times \ldots \times n$

```
int f(int n){
    if n <= 1
        return 1; // base case
    else
        return n* f(n-1);

main {
//e.g
    cout << f(4);
}
```

e.g. $f(4) = 4 \cdot f(3)$

$4 \cdot 3 \cdot f(2)$

$4 \cdot 3 \cdot 2 \cdot f(1)$

$4 \cdot 3 \cdot 2 \cdot 1$

Example: In-order

```
f(node) {
```

1) Drill down to left-most leaf (smallest value) and display value

```
    if (node.LC != NULL)
        f(node.LC)
```

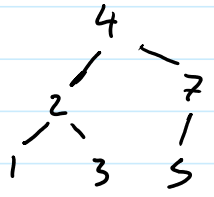
2) Display key ✓

```
    cout << node -> key
```

output;

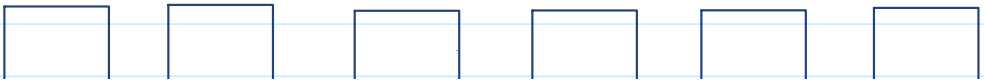
3) Drill down to right most leaf

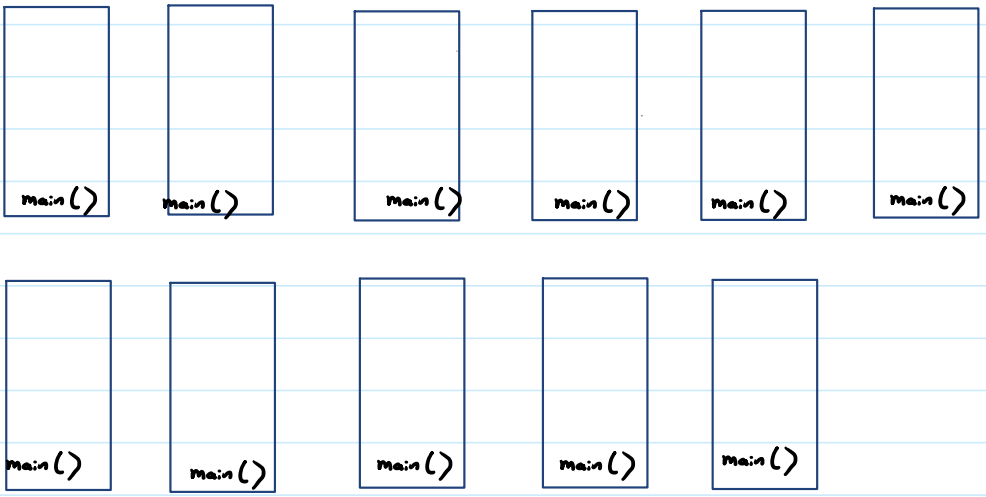
```
    if (node.RC != NULL)
        f(node.RC)
```



4) Finish execution of current instance of f()

```
}
```




main()
main()
main()
main()
main()
main()
main()
main()
main()
main()
main()