Announcement

• The same discussion topic on Friday, Monday and Wednesday.

• Active your Piazza account and enroll CSE100.

• Waitlist: section C00 is available

• Tutors lab hours are available now.
  – Check our website or
  – https://piazza.com/ucsd/fall2015/cse100/staff
1. Template

Templates are a feature of the C++ programming language that allow functions and classes to operate with generic types. This allows a function or class to work on many different data types without being rewritten for each one.

```cpp
template <class identifier> function_declaration

template <typename identifier> function_declaration
```
Template Function

• The C++ Standard Library contains the function template `max(x, y) :

```cpp
template <typename Type>
Type max(Type a, Type b) {
    return a > b ? a : b;
}
```
```cpp
// BSTNode.hpp

template<typename Data>

class BSTNode {

public:

  /** Constructor. Initialize a BSTNode with the given Data item, no parent, and no children. */
  
  BSTNode(const Data & d) : data(d) {
    left = right = parent = 0;
  }

  BSTNode<Data> * left;
  BSTNode<Data> * right;
  BSTNode<Data> * parent;
  Data const data; // the const Data in this node.

};
```
2. Operator Overloading

• In C++ the overloading principle applies not only to functions, but also to operators. That is, operators can be extended to work not just with built-in types, but also classes.
/** Pre-increment operator. */
BSTIterator<Data>& operator++() {
    curr = curr->successor();
    return *this;
}

/** Post-increment operator. */
BSTIterator<Data> operator++(int) {
    BSTIterator before = BSTIterator(curr);
    ++(*this);
    return before;
}
3. Pass by Reference vs Pass by Value

```cpp
main() {
    int i = 10, j = 20;
    swapByVal(i, j);
    cout << i << " " << j << endl;
    swapByRef(i, j);
    cout << i << " " << j << endl;
}

void swapByVal(int num1, int num2) {
    int temp = num1;
    num1 = num2;
    num2 = temp;
}

void swapByRef(int& num1, int& num2) {
    int temp = num1;
    num1 = num2;
    num2 = temp;
}
```
4. In-order Successor

• Tree traversal:
  – Pre-order
  – Post-order
  – In-order:
    • Traverse the left subtree.
    • Visit root.
    • Traverse the right subtree.
In-order traversal?

In-order: 1, 3, 4, 6, 7, 8, 10, 13, 14
Successor()

– In the BST, inorder successor of a node is the next node in inorder traversal of the BST.
How to implement it?

1. If CURRENT has a right child
   - Case 1: has a right child
     - Return the left most node in its right subtree

2. If CURRENT has no right child
   - Case 2: no right child
     - Return an ancestor that I'm in his left subtree
Questions?

Thanks