Merge homework

* M\_sort(start, mid, stop, size)
  + Inplace\_merge;
* MergeSort()
  + For (mid = l.begin(). I < size, I++)
    - Mid++;
  + M\_sort(l.begin(), mid, l.end(), l.size());
* Main
  + MergeSort(l);

Set\_bin search tree (height balanced)

* In order traversal: LNR (left node right)
  + In order(current)
    - If (current == NULL)
      * Return
    - Inorder(current-> leftChild)
    - Print(current -> value)
    - Inorder(current->rightChild)
* Template <class T>
* Void set<T>::insert(T & newElement)
  + If (count(newElement) > 0)
    - Return;
  + Node<T> \* new\_node = new node<T> (newElement, 0, 0, 0);
  + If (root == 0)
    - Root = new\_node;
  + Else
    - Root -> insert(new\_node);
* Template <class T>
* Void node<T>::insert(node<T> \* new\_node)
  + If (new\_node -> value <= value)
    - If(left\_child !=0)
      * Left\_child->insert (new\_node);
    - Else
      * New\_node->parent = this;
      * Left\_child = newNode;
  + Else
    - If (rightChild != 0)
      * Rightchild->insert(newNode);
    - Else
      * NewNode->parent = this;
      * RightChild = newNode;
* Find(x)
  + If(x == this -> value)
    - Return this;
  + If (x < this->value)
    - If (leftChild == 0)
      * Return 0;
    - else
      * LeftChild->find(x)
  + Else //must be on the right
    - If(rightChild == 0)
      * Return 0;
    - Else
      * RightChild->find(x);
* Template<class T>
* Int set<T>::size()
  + If (root == 0) return 0;
  + Else return root->size();
* Template <class T>
* Int node<T>::size()
  + Int count = 1;
  + If(leftchild != 0)
    - Count += leftChild -> size();
  + If (rightChild != 0)
    - Count += rightChild->size();
  + Return count;
* Template <class T>
* Void Set<T>::remove(node<T> \* current, T & testElement
  + If (current != 0)
    - Node<T> \* pa = current -> parent;
* Template <class T>
* Node<T> \* node<T> :: merge(node<T> \* left, node<T> \* right)
  + If (left == 0)
    - Return right;
  + If (right == 0)
    - Return left;
  + Node<T> \* child = merge(left, right -> leftChild);
  + Child->parent = right;
  + Right->leftChild = child;
  + Return right;
* Set::begin()
  + Leftslide;
* Alg to advance inorder iterator
  + At "current"