Binary Tree Housekeeping

```
// clears existing tree
template <typename Object>
void BinaryTree<Object>::clearRecurse(const NodePtr v) {
  if (v!=NULL) {
    if (v->left!=NULL) {
      clearRecurse(v->left);
      clearRecurse(v->right);
    }
    delete v;
  }
}

// clones an existing tree
template <typename Object>
typename BinaryTree<Object>::NodePtr BinaryTree<Object>::cloneRecurse(const BinaryTree<Object>& orig, const NodePtr v) {
  NodePtr n = new Node(v->element);
  if (v->left!=NULL) {
    n->left = cloneRecurse(orig,v->left);
    n->left->parent = n;
    n->right = cloneRecurse(orig,v->right);
    n->right->parent = n;
  }
  return n;
}

// Copy Constructor
template <typename Object>
BinaryTree<Object>::BinaryTree(const BinaryTree<Object>& orig) {
  sz = orig.sz;
  rt = cloneRecurse(orig,orig.rt);
}

// Overloaded Assignment Operator
template <typename Object>
BinaryTree<Object>& BinaryTree<Object>::operator=(const BinaryTree<Object>& orig) {
  if (this != &orig) {
    clearRecurse(rt);
    rt = cloneRecurse(orig,orig.rt);
  }
  return *this;
}

// Destructor
template <typename Object>
BinaryTree<Object>::~BinaryTree() {
  clearRecurse(rt);
}
```