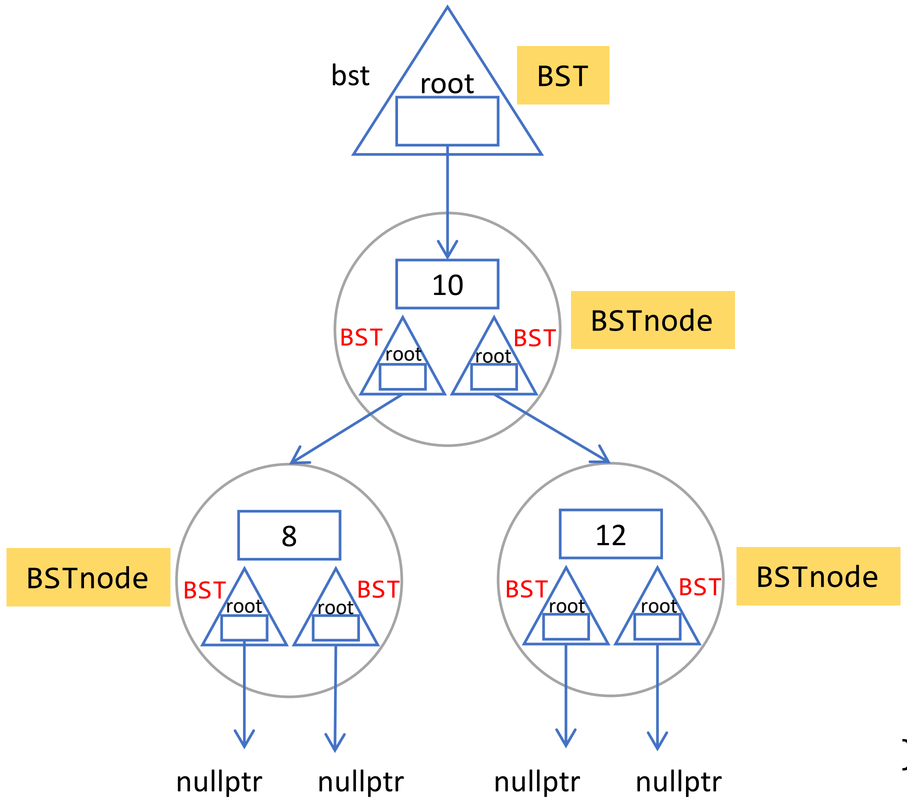
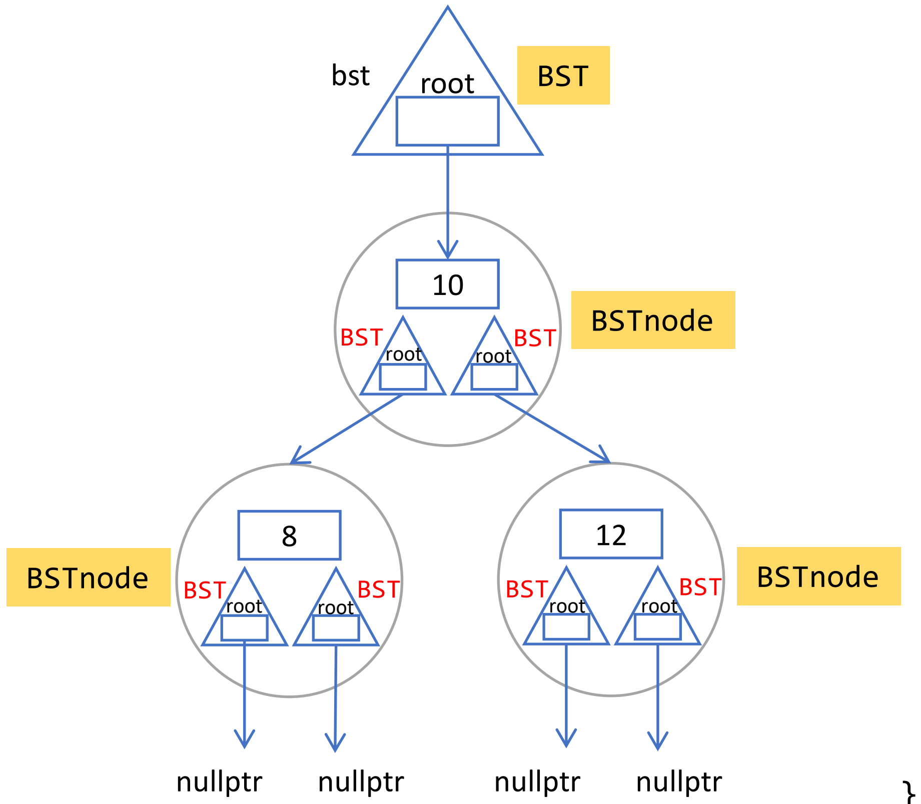
A BST which consists of 3 data



```
template <typename T>
class BST {
  private:
    struct BSTnode {
      T value;
      BST left;
      BST right;
      BSTnode(const T& x) : value(x) { }
      // BSTnode(const T& x) : value(x),
                               left(),
                               right() { }
      BSTnode(const BSTnode&) = default;
      ~BSTnode() {
        cout << "delete: " << value << endl;
        // remove right, remove left, remove value
      }
    };
    BSTnode* root = nullptr;
  public:
    BST() = default;
    ~BST() { delete root; }
  // ...
};

int main() {
  BST<int> bst;
  // ... Adding 10, 8, 12 to bst
  // Leaving main, remove bst ...
}
```

A BST which consists of 3 data



```
template <typename T>
class BST {
  private:
    struct BSTnode {
      T value;
      BST left;
      BST right;
      BSTnode(const T& x) : value(x) { }
      // BSTnode(const T& x) : value(x),
                               left(),
                               right() { }
      BSTnode(const BSTnode&) = default;
      // …
    };
    BSTnode* root = nullptr;
  public:
    BST() = default;
    ~BST() { delete root; }
    BST(const BST& bst) {
       if(bst.is_empty()) return;
       root = new BSTnode(*bst.root);
    }
  // ...
};

int main() {
  BST<int> bst;
  // ... Adding 10, 8, 12 to bst
  BST<int> bst_new = bst; // Call copy constructor
}
```