```
 1: #ifndef BINARY_SEARCH_TREE_H
 2: #define BINARY_SEARCH_TREE_H
 3:
 4: #include "Binary_Tree.h"
 5:
 6: template <typename Item_Type, typename Compare = std::less<Item_Type> >
 7: class Binary_Search_Tree : public Binary_Tree<Item_Type>
 8: {
 9: public:
10:   typedef typename Binary_Tree<Item_Type>::iterator iterator;
11:
12:   Binary_Search_Tree() { }          // initially empty tree
13:   Binary_Search_Tree(const Item_Type& item) : Binary_Tree<Item_Type>(item)  { }
14:   Binary_Search_Tree(const Binary_Search_Tree& other) : Binary_Tree<Item_Type>(other) { }
15:
16:   iterator find(const Item_Type& target) {
17:     return tracePath(target, this->getRoot());
18:   }
19:
20:   iterator insert(const Item_Type& key) {
21:     Binary_Search_Tree<Item_Type> temp(key);
22:     if (this->empty()) {
23:       *this = temp;
24:       return this->getRoot();
25:     } else {
26:       tracePath(key, this->getRoot(), false);  // sets up breadCrumb
27:       if (comp(key, *breadCrumb))
28:         return this->replaceLeftSubtree(breadCrumb, temp);
29:       else
30:         return this->replaceRightSubtree(breadCrumb, temp);
31:     }
32:   }
33:
34:   void erase(iterator pos) {
35:     if (pos.hasLeft() && pos.hasRight()) {
36:       iterator predecessor = pos;
37:       --predecessor;
38:       this->resetData(pos, *predecessor);
39:       erase(predecessor);
40:     } else {
41:       spliceOut(pos);   // has zero or one children
42:     }
43:   }
44:
45: protected:
46:   Compare comp;
47:   iterator breadCrumb;
48:   int x;
49:
50:   iterator tracePath(const Item_Type& target, iterator walk,
51:                      bool stopWhenFound = true) {
52:     bool walking = true;
53:     while (walking) {
54:       if (walk == this->end())
55:         walking = false;   // reached the bottom
56:       else {
57:         if (comp(target, *walk)) {
58:           breadCrumb = walk;
59:           walk = walk.left();
60:         } else if (comp(*walk, target)) {
61:           breadCrumb = walk;
62:           walk = walk.right();
63:         } else {  // found a match
64:           if (stopWhenFound)
65:             walking = false;
66:         }
67:       }
68:     }
69:     return walk;
70:   }
71: };
72:
73: #endif
```