```cpp
/*
Filename: p4.cpp
Author(s): Zachary Rea and Parker Ross
Date: 12 February 2023
Description: The cpp for Binary Search Tree
*/
#include <iostream>  //allows for usage of cin, cout, and cerr

#include "p4.h"

using namespace std;

//*************************************************************************
//Constructors and De-constructors
//*************************************************************************
//Constructor
//Written by Zach edited by Parker

sNode::sNode(string text) {
    this->text = text;
    left = right = NULL;
    h = 0;
}

//*************************************************************************
//Constructor
//Written by Zach

sBST::sBST() {
    treeCount = 0;
    root = NULL;
}

//*************************************************************************
//De-constructor
//Written by Zach

sBST::~sBST() {
    clear();
}

//*************************************************************************
//Private Functions
//*************************************************************************
//Function to find the minimum value of the subtree
//Written by Parker

string sBST::findMin(sNode *ptr) {
    string rc;
    if (ptr->left) {
        //look for left child
        rc = findMin(ptr->left);
    } else {
        rc = ptr->text;
    }
    return rc; //returns the minimum node text
}

//*************************************************************************
//Function for help with recursion
//Written by Zach

bool sBST::insert(sNode *&p, string text) {
    bool rc = false;
    if (p) {
        if (text > p->text) {
            rc = insert(p->right, text);
        } else if (text < p->text) {
            rc = insert(p->left, text);
```

```
70          }
71      } else {
72          p = new sNode (text);
73          treeCount++;
74          rc = true;
75      }
76      return rc;
77  }
78
79  //*************************************************************************
80  //Function for help with recursion
81  //Written by Zach edited by Parker
82
83  bool sBST::remove(sNode *&p, string text) {
84      bool rc = false;
85      if (p) {
86          //case for the correct node
87          if (text == p->text) {
88              //recursively find the min of right child and replace
89              if (p->right) {
90                  p->text = findMin(p->right);
91                  rc = remove(p->right,p->text);
92                  //simply replace with left child if no right child
93              } else if (p->left) {
94                  sNode *t = p;
95                  p = p->left;
96                  delete t;
97                  treeCount--;
98                  rc = true;
99                  //no children just delete
100             } else {
101                 delete p;
102                 treeCount--;
103                 rc = true;
104             }
105             //recursion for children
106         } else if (text < p->text) {
107             rc = remove(p->left,text);
108         } else if (text > p->text) {
109             rc = remove(p->right,text);
110         }
111     }
112     return rc;
113 }
114
115 //*************************************************************************
116 //Function for help with recursion
117 //Written by Parker
118 bool sBST::isIn(sNode *p, string text) const {
119     bool rc = false;
120     if (p){
121         if (text < p->text) {
122             rc = isIn(p->left,text);
123         } else if (text > p->text) {
124             rc = isIn (p-> right,text);
125         } else {
126         rc = true;
127         }
128     }
129     return rc;
130 }
131
132 //*************************************************************************
133 //Function for help with recursion
134 //Written by Parker
135
136 void sBST::printIt(sNode *p) const{
137     if (p){ //making sure function has a printable node
138     // prints left subtree, root then right subtree for ascending order
```

```
139         printIt(p->left); //prints Left subtree
140         cout << p->text << endl; //print root
141         printIt(p->right); //prints right subtree
142     }
143 }
144
145 //*************************************************************************
146 //Function for help with recursion
147 //Written by Zach
148
149 void sBST::clear(sNode *p) {
150     if (p) {
151         clear(p->left);
152         clear(p->right);
153         delete p;
154     }
155 }
156
157 //*************************************************************************
158 //Public Functions
159 //*************************************************************************
160 //Function to insert the text into the tree
161 //Written by Zach
162
163 bool sBST::insert(string text) {
164     return (insert(root, text));
165 }
166
167 //*************************************************************************
168 //Function to remove the node with the given text
169 //Written by Zach
170
171 bool sBST::remove(string text) {
172     return (remove(root, text));
173 }
174
175 //*************************************************************************
176 //Function to tell if the tree co`ntains the given text
177 //Written by Parker
178
179 bool sBST::isIn(string text) const {
180 return (isIn(root, text));
181 }
182
183 //*************************************************************************
184 //Function to print the BST values in ascending order
185 //Written by Parker
186
187 void sBST::printIt() const{
188     printIt(root);
189 }
190
191 //*************************************************************************
192 //Function to show the number of nodes in the tree
193 //Written by Zach
194
195 int sBST::count() const{
196     return treeCount;
197 }
198
199 //*************************************************************************
200 //Function to remove all of the nodes in the tree
201 //Written by Zach
202
203 void sBST::clear() {
204     clear(root);
205     treeCount = 0;
206     root = NULL;
207 }
```

```
208
209   //Dr. Wheat, I understand that our code is seg faulting when the program
210   //attempts to remove a node. Even after stubbing findMin, we were unable
211   //to find the cause of the problem when we implemented the p4a corrections.
```