Salazar

Assignment 6

Assignment 6 (Binary search tree)(two week assignment)

Completion requirements

Create a design, before you start coding, that shows how your binary tree functions and what attributes it keeps track of to function (yes, you can add to this design once you start coding, but please get some design down to start with and make note of when you add new design features based on your implementation work🙂),

Create some tests (at least one per function), before you start coding, that you want your Binary Search Tree (BST) to pass as evidence that it would be working correctly if it passed the tests,

Implement a binary search tree that includes:

nodes to store values,

an add function that adds a new value in the appropriate location based on our ordering rules,

(I likely used less than or equal to going to the left and greater than values going to the right)

a remove function that finds and removes a value and then picks an appropriate replacement node,

(successor is a term often used for this)

we have at least one tree traversal function (I recommend starting with an in-order traversal!)

Bonus if you implement the three common traversals (pre-order, post-order, in-order)

More Bonus if you also include a breadth-first traversal (sometimes called a level-order search)

Analyze and compare the complexity of insert and search as compared to a binary tree without any order in its nodes (what is the run-time of an unordered tree...?).

Once you have implemented and tested your code, add to the README file what line(s) of code or inputs and outputs show your work meeting each of the above requirements (or better, include a small screen snip of where it meets the requirement!).

//Desigin

// node struture

   // sauce: a string for the hot sauce

   // hotness: a value for the hot sauce hotness , changed to shu

   // node pointers for back and front.  // front = left and back = right,

// start with my base from assigment 3 then made lots of changes from there

// add function to place sauces into tree

// remove function delete elements

// a search function

// a travel function that start from the left to the right.

//requirements

//an add function that adds a new value in the appropriate location based on our ordering rules,

A screen shot of a computer code

Description automatically generated

//traverse function

A screen shot of a computer code

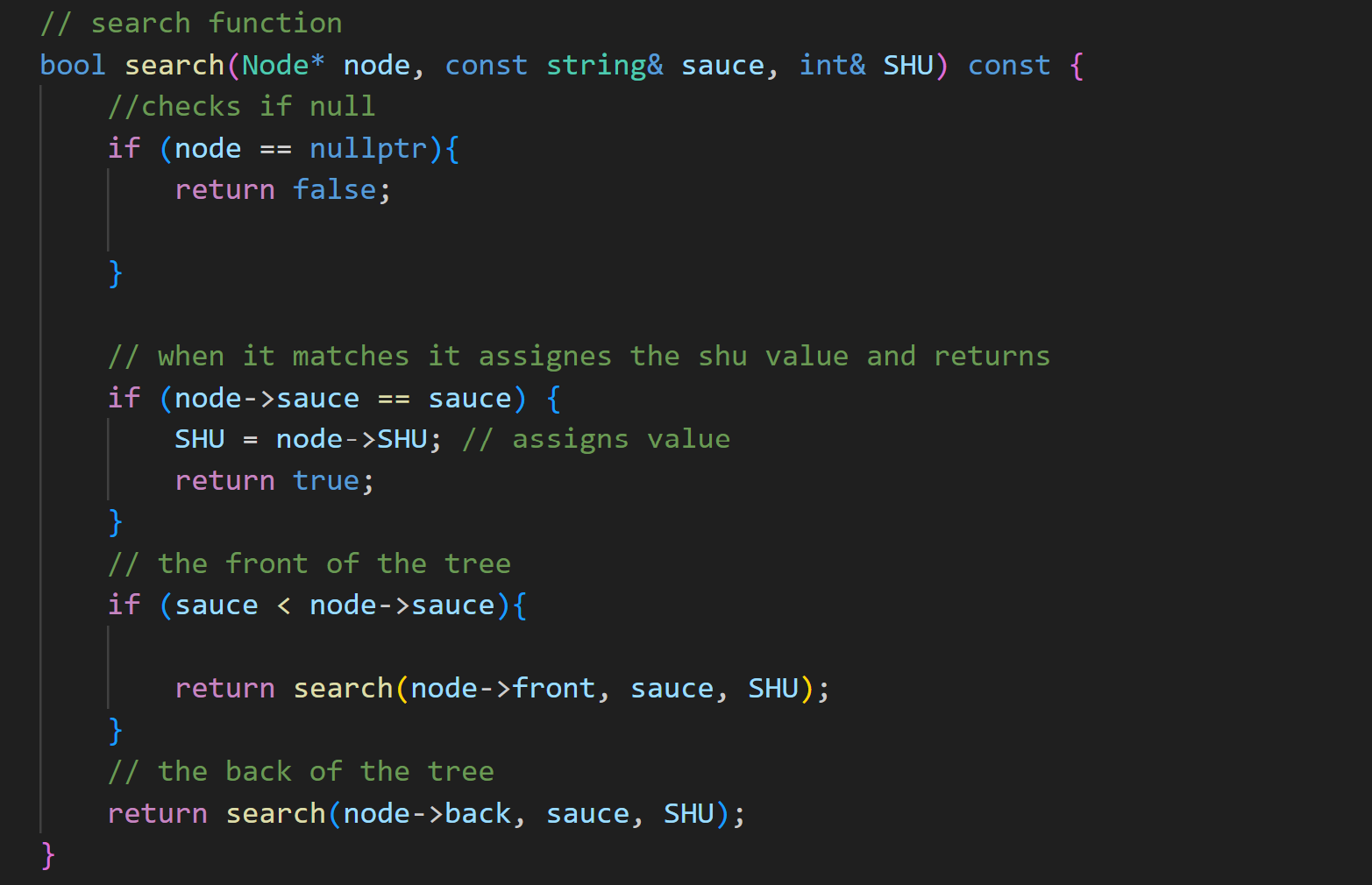
Description automatically generated

// remove function

A screenshot of a computer program

Description automatically generated

// search function



// test functions

A computer screen shot of a program

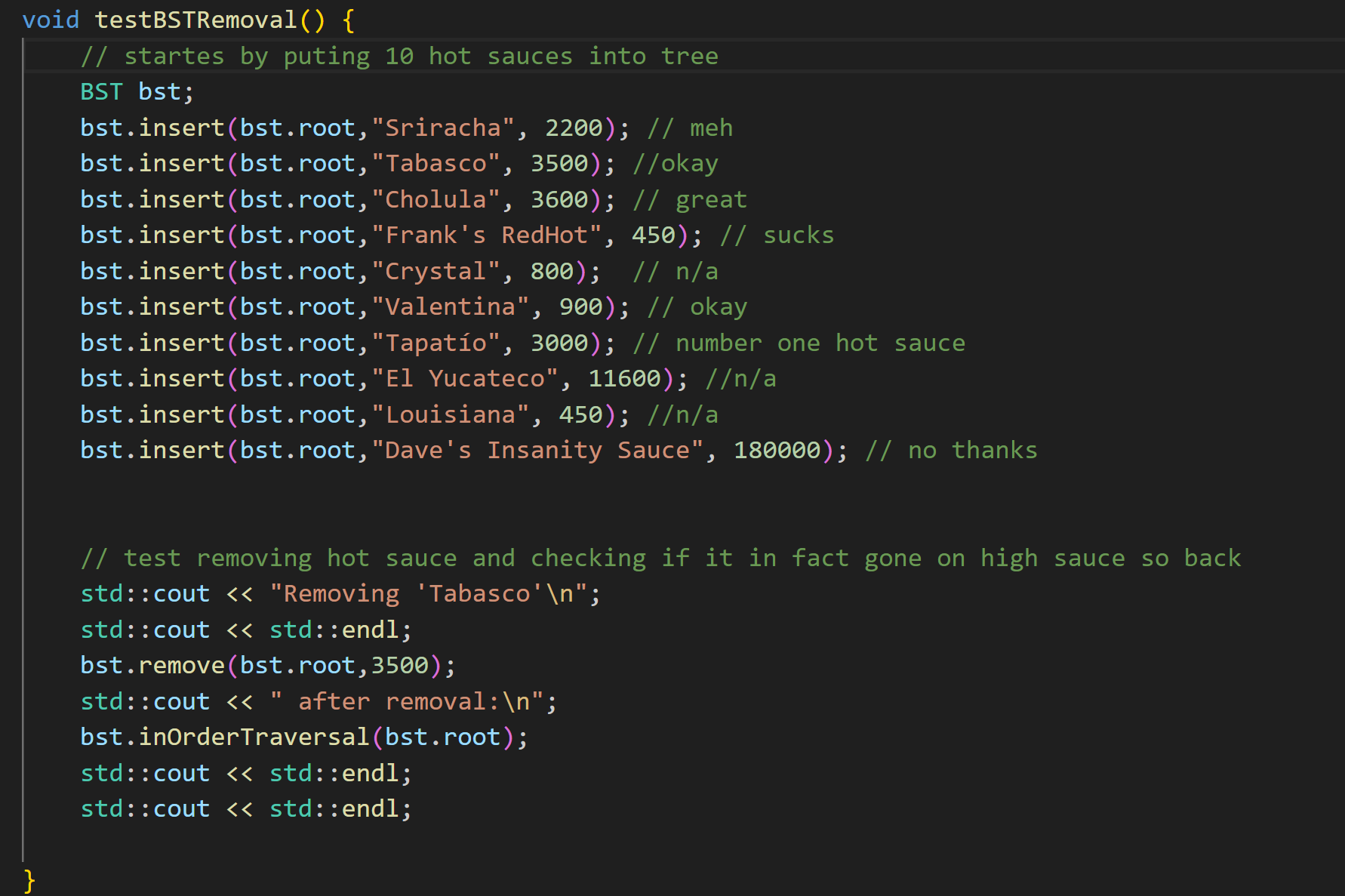
Description automatically generated

//search test

A computer screen shot of text

Description automatically generated

//removal test



// test results

A black screen with white text

Description automatically generated