Assignment Objective: Build skills on C class creation and integration while implementing a **Binary Search Tree (BST)**.

Requirements:

* Create a class called sNode with the following members
  + private:
    - string text; // holds the node’s key
    - int h; // for future use
    - sNode \*left, \*right; // left and right children pointers
    - The constructor for sNode, which should be of the form sNode(string text = “”)
* Create a class called sBST with the following members
  + Private members:
    - An integer “treeCount” to record the number of entries in the BST
    - An sNode \*root; to point to the tree of nodes.
    - string findMin(sNode \*ptr) which returns the minimum value in the subtree whose root node is pointed to by ptr.
    - Private methods to support insert(), remove(), isIn(), printIt(), and clear().
  + Public members:
    - constructor sBST() that causes the object to be initialized
    - destructor ~sBST() –deletes all the nodes in the BST in preparation for the BST to be destroyed.
    - bool insert(string text) – Inserts the text into the tree; returns true for success; returns false if the text value is already in the tree.
    - bool remove(string text) – removes the node with that text, deleting the node; returns true on success; returns false if the node is not found.
    - bool isIn(string text) const – returns true if the node with text value is in the tree; otherwise returns false.
    - void printIt() const – results in the BST’s text values being printed in ascending order, one value per line.
    - int count() const – returns the number of nodes in the tree
    - void clear() – removes all nodes from the BST, making the BST empty.
* You must not use any other data structure, whether built-in or otherwise.
* Compile your program: g++ p4.cpp p4m.cpp -o p4
* First submission: The highlighted portions shall be submitted as described below. Note, to do any testing, there should be a “stub” for each of the functions that are not yet completed.
* Final submission: The totality of the assignment shall be submitted as described below.
* Demonstrate that the sBST data structure works:
  + Run your program as follows:

p4 p4In\_1.txt p4Remove\_1.txt p4IsIn\_1.txt > p4output1.txt

p4 p4In\_2.txt p4Remove\_2.txt p4IsIn\_2.txt > p4output2.txt

* + Compare your output files to the corresponding p4CorrectOutput1.txt and p4CorrectOutput2.txt files.
* Deliverables:
  + Turned into D2L: put a zip file containing:
    - Your p4.h file
    - Your p4.cpp file
    - Your p4output1.txt file
    - Your p4output2.txt file
    - DO NOT put a project into D2L
  + Turned into class: a hardcopy of the above files in the order given.