Declare and implement a class that provides the following interface
EXACTLY.
You will modify your tree files, bs_tree.h and bs_tree.cpp
You will also need to use your bst_node.h and bst_node.cpp from the
previous exercises.

Class Name: BSTree

Private Members:

  bool Insert(int, BSTNode*&)
  --creates a new BSTNode, inserts it into the tree, and returns true
    if the integer is already in the tree, does not insert, and
returns false

  void Clear(BSTNode*&)
  --clears the entire contents of the tree,
    freeing all memory associated with all nodes

  string InOrder(BSTNode*)
  --creates a string of the data in all nodes in the tree, in
ascending order
    separate by spaces (there should be a space after the last output
value)

  bool Remove(int, BSTNode*&) (*NEW*)
  --traverses the tree and removes the node containing the target
    integer if present and returns true
    return false if target integer is not in tree (or the tree is
empty)

  int FindMin(BSTNode*) const (*NEW*)
  --returns the value of the smallest node in the tree
    helper function for private Remove()

  BSTNode* root_
  --points to the root node of a binary search tree
  unsigned int size_
  --holds the number of nodes in the tree

Public Members:

  Default Constructor
  --sets root_ to NULL
    sets size_ to 0

  Destructor
  --calls Clear()

  bool Insert(int)
  --calls private function Insert(int, root)

```
  bool Remove(int) (*NEW*)
  --returns value returned by private function Remove(int, root)

  int FindMin() (*NEW*)
  --if the tree is empty return 0
    otherwise return the value returned by private function
FindMin(root)

  void Clear()
  --calls private function Clear(root)

  unsigned int size() const
  --Accessor for size_

  string InOrder()
  --calls private function InOrder(root)

A Makefile has been included with this exercise.
Type make to see available options

Please note you need to follow programming style guidelines for this
exercise.

Labs having the following will lose 10% per category:
- Code (such as output statements) not explicitly required
- Bad Programming Style (Variable Names, Formatting, Fails Style
Check)

Labs that DO NOT COMPILE will receive a ZERO!

NOTE: Labs that compile with warnings will be treated as if they did
not
      compile. (i.e. they will receive a zero)
```