Data Structures	Assignment 3
CCO 2103-01, Spring 2024	Due at 5:00pm, Wednesday, May 1

Hand in your solutions electronically using LearnUs. As was announced in class, you are not allowed to use any data structure libraries, including the linked lists, stacks, queues, trees, binary search trees, priority queues, and heaps provided by JDK. When in doubt, contact the course staff.

This assignment is *not* a completion points assignment.

Submit your source code(s), zipped as `yourStudentID.zip`. For example, if your student ID is `2023000000`, then you must zip all your source code(s) into `2023000000.zip` and submit this file. Each class should have its own `.java` file, of which the filename is the same as the class name. Do *not* include your student ID as part of the class names.

This assignment consists of one programming task.

(1) (100 points) Write a class that implements a binary search tree of integers.

Your code must have the following two classes: `BST` and `BSTNode`. The `BSTNode` class represents a node in a binary search tree, and must have the following fields:

- `public int key;`
  Holds the key value of the node.
- `public BSTNode left, right;`
  Each holds the reference to the left and the right child, respectively. Is `null` if the corresponding child does not exist.

It is important to declare these fields (and what follows) as `public`, because our grading program will directly access these fields and methods to check if they are correctly set. Follow the convention from class that the left subtree of any given node x contains smaller key values than that of x.

The `BST` class is the main class that implements a binary search tree. It must have the following fields and methods:

- `public BSTNode root;`
  Holds the reference to the root node. Is `null` if the tree is empty.
- `public BST();`
  Constructs an empty binary search tree.
- `public void insert(int key);`
  If there exists a node whose key value is `key`, does nothing. Otherwise, inserts a new node whose key value is `key`.
- `public boolean search(int key);`
  If there exists a node whose key value is `key`, returns `true`; otherwise, returns `false`.

- `public boolean delete(int key);`
  If there exists a node whose key value is `key`, deletes it and returns `true`. Otherwise, does nothing and returns `false`.

You must use the algorithms presented in class. The resulting tree must be *exactly* what those algorithms would return. In particular, when you insert a node, the rest of the tree must remain the same, except for the single newly created node. Likewise, when you delete a node, the height of the tree must not increase.

You are free to add your own methods and fields as you see fit, but you need to implement all of those methods. You do not need to provide an entry point to your code. Our grading program will use your class and check if your code works correctly. (Of course, you would need to write your own driver that uses your class to test your code before you submit your solution.)