Data Structures	Completion Points Assignment 4
CCO 2103-01, Spring 2024	Due at 5:00pm, Saturday, June 22

Hand in your solutions electronically using LearnUs.

This assignment is a completion points assignment: any reasonable attempt will receive *full* points, *even if it is incorrect*.

This is a programming 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. You are not allowed to use any data structure libraries provided by JDK.

Try to solve this problem without referring to the lecture notes. The problem description starts on the next page.

(1) (10 points) Write a class that implements a red-black tree. The key values of the tree are integers, and each integer is associated with a string "payload."

Your code must have the following two classes: `Node` and `RBT`. The `Node` class represents a node in a red-black tree, and must have the following fields:

- `public boolean red;`
  Is `true` if the node is red; `false` if black.
- `public int key;`
  Holds the key value of the node.
- `public String payload;`
  Holds the payload of the node.

You are free to add your own methods and fields as you see fit, as long as you satisfy the given specifications.

The `RBT` class implements a red-black tree. It must have the following fields and methods:

- `public RBT();`
  Constructs a new empty red-black tree.
- `public Node root;`
  Holds the reference to the root node.
- `public boolean isEmpty();`
  Returns `true` if the tree is empty; `false` otherwise.
- `public boolean hasLeftChild(Node node);`
  Returns `true` if `node` has a non-null left child. Returns `false` if the left child of `node` is a "black null leaf." You can assume that `node` is always given as an existing node of the tree.
- `public boolean hasRightChild(Node node);`
  Returns `true` if `node` has a non-null right child. Returns `false` if the right child of `node` is a "black null leaf." You can assume that `node` is always given as an existing node of the tree.
- `public boolean insert(int k, String p);`
  If `k` already exists in the tree, does nothing and returns `false`. Otherwise, `k`, along with the payload `p`, is inserted to the tree and `true` is returned.
- `public boolean delete(int k);`
  If `k` exists in the tree, deletes the node and returns `true`. Otherwise, does nothing and returns `false`.
- `public String query(int k);`
  If `k` exists in the tree, returns the associated payload. Otherwise, returns `null`.

Submit all your source codes, including `Node.java`, `RBT.java`, and any other source files you created.