ASSIGNMENT 3: B-TREES with DUPLICATES

Goal: The goal of this assignment is to get some practice with binary search trees, specifically B-Trees. B-Trees are one of the most important data structures we will study in this class – they are regularly used in large database systems for storing indexes on the records. However, just for fun, we will make one significant departure from standard B-trees. We will allow duplicate keys, i.e., a tree can have an unbounded number of keys that have the same value.

Problem Statement: Your task is to implement a generic DuplicateBTree class. As is to be expected, DuplicateBTree must implement the BTree ADT we studied in class, except that it can contain multiple copies of the same key (at times with exactly the same value). This implies you will have to change the algorithms for searching, inserting and deleting suitably. The exact interface will be as follows:

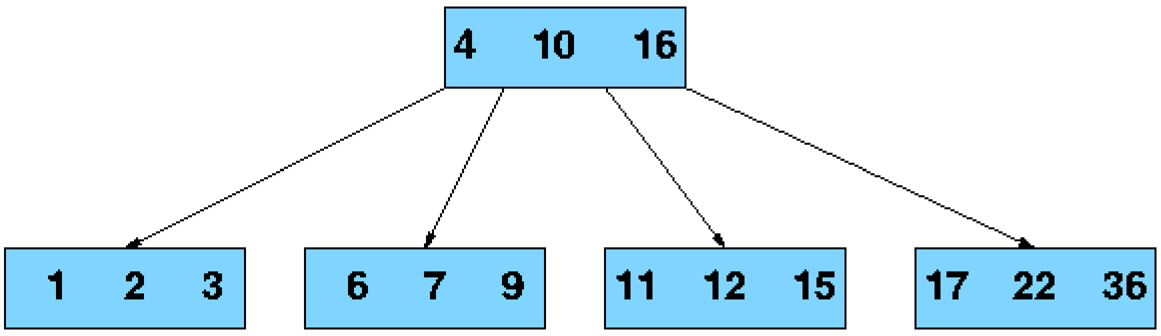
public interface DuplicateBTree<Key, Value> {

```
        public BTree(int b) throws bNotEvenException;  /* Initializes an empty b-tree.
Assume b is even. */
        public boolean isEmpty();  /* Returns true if the tree is empty. */
        public int size();  /* Returns the number of key-value pairs */
        public int height();  /* Returns the height of this B-tree */
        public Vector<Value> search(Key key) throws IllegalKeyException; /* Returns all
       values associated with a given key in a vector */
        public void insert(Key key, Value val);  /* Inserts the key-value pair */
        public void delete(Key key) throws IllegalKeyException;  /* Deletes all
       occurrences of key */
        public String toString(); /* Prints all the tree in the format listed below */
```

}

Your implementation will have to define a node class which will maintain all keys in the node (you may use an array for this), and references to all children. The format for toString function defined above is as follows:

To convert a tree to string, just convert its root to string. Converting a node to string is defined recursively. Nodes that are null, are mapped to empty strings. The mapping of a node, is just the concatenation of the mappings of its children, separated by the key value pairs present at the node, and surrounded by rectangular brackets []. For example, neglecting the values,



will be converted as:

”[[1, 2, 3], 4, [6, 7, 9], 10, [11, 12, 15], 16, [17, 22, 36]]”

Now, we have key value pairs instead of simple set of keys, we just replace a single key output in the above string by the key value pair. For example, if in the above tree, the values were the string representations of the numbers, then the answer would be:
”[[1=One, 2=Two, 3=Three], 4=Four, [........”

What is being provided?

We have provided code that can be used to interact with your btree implementation using a simple commands. We expect you to not alter that code, as it will be used to check the correctness of your code. You can only modify the BTree.java file, along with adding new java classes.

The provided folder is a valid eclipse and intellij idea project, and can be directly imported or opened using the IDE of your choice. You can also work using simple text editors like vim if you want.

What to submit?

1. Submit your code in a .zip file named in the format **<EntryNo>.zip.** Make sure that when we run “unzip yourfile.zip” in addition to your code “writeup.txt” should be produced in the working directory. Also, an executable jar file must be created in the same location as the writeup.txt, and your code must be present in the same directory structure as that is given in the provided code.
   In summary, on unzipping, three things must be created in the current work directory
   COL106_A3 directory, that contains your modified code
   writeup.txt
   <EntryNo>.jar the executable jar file

   You will be penalized for any submissions that do not conform to this requirement.

2. The writeup.txt should have a line that lists names of all students you discussed/collaborated with (see guidelines on collaboration vs. cheating on the course home page). If you never discussed the assignment with anyone say None.
   After this line, you are welcome to write something about your code, though this is not necessary.

Evaluation Criteria

The assignment is worth 5 points. Your code will be autograded at the demo time against a series of tests. Three points will be provided for correct output and two points will be provided for your explanations of your code and your answering demo questions appropriately.

What is allowed? What is not?

1. This is an individual assignment.
2. Your code must completely be your own. You are not to take guidance from any general purpose code or problem specific code meant to solve these or related problems. Remember, it is easy to detect this kind of plagiarism.

3. You are not allowed to use built-in (or anyone else's) implementations of trees, nodes or other basic data structures. A key aspect of the course is to have you learn how to implement these data structures. You are, however, required to use Java's built in java.Util.Vector class.
4. You should develop your algorithm using your own efforts. You should not Google search for direct solutions to this assignment. However, you are welcome to Google search for generic Java-related syntax.
5. You must not discuss this assignment with anyone outside the class. **Make sure you mention the names in your write-up in case you discuss with anyone from within the class.** Please read academic integrity guidelines on the course home page and follow them carefully.
6. Your submitted code will be automatically evaluated against another set of benchmark problems. You get significant penalty if your output is not automatically parsable and does not follow input-guidelines.
7. We will run plagiarism detection software. Anyone found guilty will be awarded a suitable penalty as per IIT rules.