Assignment 5
B-Trees

Release Time	Due Date
11/15/2017	**11/30/2017**

Objectives

- To practice building and searching B-tree.
- Practice developing high-performance solutions.

Problem Specification

In computer science, ASCII table is very important. For this assignment, you need to implement a tool to search an ASCII table using B-trees as discussed in class (i.e., actual keys and record pointers at leaves, the values at interior nodes help us search/insert/delete, etc). In order to make coding a little bit easier – keep three values at every interior-node for a 2-3 tree or four values for a 2-3-4 tree.

1) Read the information from the website below to understand ASCII table.

   http://www.asciitable.com/

2) Build some 2-3 tree indexes for this ASCII table. You can build a tree based on each column, i.e., the values in that column act as primary keys. (Ignore the extended ASCII codes and the html column.)

3) In your tool, you should ask user to give an input, and print out relevant information about that input.

   a. Your program should recognize the format of input automatically. For example, if the input is BS then it is a char. You need to print the Dec, Hx and Oct information "Dec: 8, Hx: 8, Oct: 010."

   b. If the input is arbitrary, you should print out all the possible results. Consider A: 1. If A is Hx, Dec:10, Oct: 012, Char: LF, 2. If A is Char, Dec: 65, Hx:41, Otc:101.

4) You must design and implement all the data structures in your program. In other words, you cannot use built-in data structures that are readily available now a days in many high-level languages.

5) Conduct level-by-level traversal and print your traversal result. For each non-leaf node, print the three keys (i.e., max(leftSubtree), max(middleSubtree), max(rightSubtree)), the keys of the parent node, and the keys of its children nodes. For leaf-nodes, print the whole record.

6) Re-do steps (2)-(5) with a 2-3-4 tree. Remember to keep the trees of min-height.

7) Please compare the differences between your implementations of 2-3 and 2-3-4 trees, you need to provide the information about height, number of leaves and searching time.

NOTE: Since ASCII table is not large, there are other solutions (e.g. array lookup or hashing based) that are more efficient for searching ASCII table, but this assignment is for you to learn B-tree coding.

Design Requirements

Code Documentation

For this assignment, you must include documentation for your code as generated by JavaDoc. You should have JavaDoc comments for every class, constructor, and method. By default, JavaDoc should output html documentation to a subfolder within your project (/dist/javadoc). Make sure this folder is included when you zip your files for submission. You do not need to submit a hard copy of this documentation.

Hint: http://stackoverflow.com/questions/4468669/how-to-generate-javadoc-html-in-eclipse

Coding Standards

You must adhere to all conventions in the CS 3310 Java coding standard. This includes the use of white spaces for readability and the use of comments to explain the meaning of various methods and attributes. Be sure to follow the conventions for naming classes, variables, method parameters and methods.

Testing

Make sure you test your application with several different search values, to make sure it works.

Assignment Submission

- Generate a .zip file that contains all your files, including:
    - Source code files
    - Including any input or output files
    - Documentation of your code – e.g. using Javadoc if using Java
    - A brief 1-2 paragraph report (in a pdf or txt file) on your comparison.

Don’t forget to follow the naming convention specified for submitting assignments