**COP3530 Data Structures and Algorithms**

**Project 2: Implement a B Tree**

**due 2/13/15**

**Brief Description:**

Develop a B tree. The Tree should hold string keys and string values. The size of each node should be taken in as a parameter and not fixed.

Your B Tree shall implement the following functions.

* Insert – Takes in a key-value pair and inserts it into the tree.
* Delete – Deletes a key from the tree.
* Find – Takes in a key and searches for it in the tree. If found, it returns true else it returns false. If found, it store the value in the second parameter passed to it.
* toStr – concatenates the contents of the tree from an inorder traversal into a return string with each string item followed by a newline character in the return string – for example, if the tree had insert(“alpha”,“a”), then insert(“beta”,“b”), then insert(“gamma”,“g”), the toStr method should produce “alpha\nbeta\ngamma\n” as the returned string.
* appropriate constructors and destructors

**Deliverables**

* Well commented and well-structured source code. Your header file shall be named bTree.h
* Makefile that compiles your code with our bTree\_main.cpp on the CISE linux machines and produces executable bTree. We will provide the main file. The program must run on sand.cise.ufl.edu, so be sure to test it there (use remote access as required).
* PDF document containing a description of code organization, any special diagnostics, test cases and test results, as well as known bugs. This shall be named according to usual convention.

**Interface**

class bTree {

private:

 //Your private stuff to implement the B Tree

public:

 //Constructor

 bTree();

 //Destructor

 ~bTree();

 //Inserts the value pair into the tree

 bool insert(string key, string value);

 // Searches for the key in the tree, if found, it returns

 // true else returns false.

 // If the key is found, it stores the value in the \*value variable

 bool find(string key, string \*value);

 //Searches for the key and deletes it. Returns true if deletion was

 //successful, returns false if the key wasn't found

 bool delete\_key(string key);

 // concatenates the contents of the tree from an inorder traversal

 // into an output stream with each string item followed by a

 // newline character in the stream

 string toStr();

};

You may want to also provide helper methods (of course), as well as other public methods that are not included in this interface in order to help you develop/diagnose/debug your class.

**Grading Rubric for Part a**

|  |  |
| --- | --- |
| **Criterion** | **Points** |
| Code compiles without error from makefile | 5 |
| Code efficiently gives correct answers to test cases | 50 |
| Testing (Writing appropriate test cases and testing the code) | 15 |
| Proper code comments | 10 |
| Good code documentation (structure, how to use) | 10 |
| Following good programming practices | 10 |
| Total | 100 |

Remember, ALL of your submissions MUST compile on **sand.cise.ufl.edu using your makefile**. You may develop your code on your own system/environment, but upload it to your CISE account and test it on the CISE machines well before you submit it. Code that does not compile on the CISE machines from the makefile WILL BE RETURNED ungraded and points (see rubric) will be deducted, as we will NOT debug or port either you code or your makefile.