DATA STRUCTURES AND ALGORITHMS ASSIGNMENT

Task 1 – Binary Search Tree – 6 Marks Implement a binary search tree using char as the data records. First write a suitable structure to represent a node. Create a node pointer in your main function to represent the root of a tree. Next implement the following functions, and test them from main():

1. void tree\_insert ( Tree\_Node\*\* root, char data ); Creates a new node in the tree in the appropriate position for a binary search tree.

2. Tree\_Node\* tree\_search ( Tree\_Node\* root, char data ); If a value exists in the tree, return a pointer to that node, otherwise return NULL.

3. void tree\_print\_sorted ( Tree\_Node\* root ); Traverse the tree, printing the data held in every node, in smallest-to-greatest sorted order.

4. void tree\_delete ( Tree\_Node\* root ); Delete every node in the tree without creating a memory leak. Your input string should not be obtained from the user. It is sufficient to load the following sequence of characters into the tree and print the output in sorted order as shown below.

ADDING: F L O C C I N A U C I N I H I L I P I L I F I C A T I O N

SORTED: A A C C C C F F H I I I I I I I I I L L L N N N O O P T U C in tree Z not in tree

OUTPUT:

A A C C C C F F H I I I I I I I I I L L L N N N O O P T U

C in tree

C in tree

C in tree

C in tree

Z not in tree

Task 2 – A Practical Application – 4 Marks You have been hired to build a reasonably simple database which stores documents in memory. This database will be used by a small text-based search engine to curate books that users might want to read. The database should store the name and word count of each book. It should also assign a unique identifier to each book as it is added to the database. Because users can ask complex queries the database needs to be extremely fast. Proper choice of datastructure will have a big impact on how well the system performs. The developer who was hired before you attempted to build the database using a linked list – a poor design decision by all accounts. Aforementioned developer was summarily sacked. You, being that much smarter, have opted to use a Binary Search Tree for your database. If it is well implemented then it should perform drastically better than the linked list. Your employers know what they are looking for and have already provided tools to assess the performance of your implementation. All of the testing code you need is in place. All you need to do is fill out the bstdb.c file in the src directory. Specifically, you need to write six functions:

1. int bstdb\_init ( void ); Allocates any memory that your database will need (e.g. if you need to allocate some initial space for your binary search tree). Should return 0 if initialization fails for some reason

2. int bstdb\_add ( char \*name, int word\_count ); Add a new book to your binary search tree. Function should store the name and word count, then return a unique ID which can be used to retrieve the book later.

3. int bstdb\_get\_word\_count ( int doc\_id ); Retrieves the word count of a book from the database using its document ID.

4. char\* bstdb\_get\_name ( int doc\_id ); Retrieves the name of a book from the database using its document ID.

5. void bstdb\_stat ( void ); Use this function to run some tests of your own. It will be called once by the profiler after it has run its own tests. For example, you might use this function to show that your binary search tree is balanced. The linked list developer tested whether or not the linked list was storing as many books as it claimed. They also computed the average number of node visited when retrieving a result. These values should be printed to the terminal.

6. void bstdb\_quit ( void ); Release any dynamically allocated memory that your BST used. The bstdb.c source file for the project has been heavily commented to explain what each function should do. You have also been provided with the previous developer’s attempt in the src/db/listdb.c file. If you are looking for some form of guidance regarding how to proceed you might want to check those files.