BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI
Second Semester 2013-2014
CS / IS F211 Data Structures and Algorithms
Lab Sheet 6

Problem Statement :

Consider an application for maintaining list of books in a library. It should store all the details such as book accession number, title, author name and price. Construct a binary search tree (BST) to maintain a list of books using linked list. Let the key in BST be accession number. Application should be able to perform operations such as insert a book record, find a book, delete a book record from BST, find a book with maximum and minimum price etc. Consider accession number of the book as key for performing all the operations.

Use following data structures:

(1) BST : Construct a structure BST (Binary Search Tree) which includes pointer to root node of binary search tree, total number of books in library and height of the tree.

(2) book : Construct a structure book which includes details of book such as accession number (unique key, declare it as integer), title, author name (consider only first author) and price.

(3) book_node: Construct a structure book_node which includes an instance of book structure, pointer to left child and pointer to right child in BST.

Write the following functions to perform the above mentioned operations:

(1) `BST createEmptyBST()`
This function creates an empty BST, initializes its members and returns the created BST.

(2) `BST insert_BST(BST bt, book b)`
This function inserts a book record (according to accession number as key) at the appropriate position in BST and returns updated BST.

(3) `boolean find_BST(BST bt, book b)`
This function searches for a book record with accession number as key and returns true if found, returns false otherwise.

(4) `BST delete_BST(BST bt, book b)`
This function deletes the book record corresponding to key from BST and returns the updated tree. Consider all the cases of deletion in BST.

(5) `book find_MaxPrice(BST bt)`

This function visits the nodes of BST in in-order fashion and finds the book with maximum price. It returns the book record with maximum price.

(6) `book find_MinPrice(BST bt)`

This function visits the nodes of BST in pre-order fashion and finds the book with minimum price. It returns the book record with minimum price.

(7) `int getHeight(BST bt)`

This function computes and returns the height of BST.

(8) `book find_latestBook(BST bt)`

This function traverses BST, searches a book with maximum accession number and returns the book record. Book with maximum accession number is latest book in library.

(9) `book find_oldestBook(BST bt)`

This function traverses BST, searches a book with minimum accession number and returns the book record. Book with minimum accession number is oldest book in library.

(10) `BST reOrderBST(BST bt)`

This function traverses BST in in-order fashion. It removes each node from BST bt and inserts that node in a new BST with price as a key. It returns the new BST. If more than one book has same price then it should make it as left child.

Exercise: Modify the above function `reOrderBST(BST bt)`. Here if more than one book has same price then create a linked list of the books with same price at that corresponding node instead of make it as a left child.

Write a separate driver.c file for invoking all the functions. Generate input.txt file with 10 to 15 book records manually for testing your program.

Deliverables: Driver.c, BSTDef.h, BSTOps.h and BSTOps.c.

****