A2 Computer Science

Binary Search Tree ADT

Introduction

See here for a general introduction to binary trees:

https://isaaccomputerscience.org/concepts/dsa_search_bst?examBoard=all&stage=all

Binary Search Trees

Binary Search Trees are binary trees that do not contain duplicate Nodes.

A Binary Search Tree is defined as follows:

- The left subtree of a node contains only nodes with keys **less than** the node's key.
- The right subtree of a node contains only nodes with keys **greater than** the node's key.
- Both the left and right subtrees must also be binary search trees.

See here for how to implement a binary search tree as an array of records:

https://isaaccomputerscience.org/concepts/dsa_datastruct_tree_implementation?examBoard=all&stage=all

See here for a description of tree traversal techniques
https://isaaccomputerscience.org/concepts/dsa_datastruct_tree_traversals?examBoard=all&stage=all

Task

1. Create a class for a binary search tree ADT

2. Create a program that uses the class to:
   - add data to the tree
   - search the tree for data
   - perform the three types of tree traversal (in order, pre order and post order)

Hints:

a) Create a class to implement a Node object.

The object should have two attributes:
data - to hold a data item
leftPtr – to point to the left branch of the tree
rightPtr – to point to the right branch of the tree

b) Create a class to implement a binary search tree ADT

Implement your binary tree ADT as a fixed size list (array) of node objects

The Tree class should include the storage for the tree and the following methods:

Method	Description
__init()__	The constructor. Creates the store for the tree and performs any other initialisation needed. The store size is set by an argument to the constructor.
insertItem(newItem)	Inserts a data item into the tree.
findItem(searchItem)	Finds the specified item in the tree.
inOrder()	Performs an “in order” traversal to display the entire tree.
preOrder()	Performs a “pre order” traversal to display the entire tree.
postOrder()	Performs a “post order” traversal to display the entire tree.
displayStore()	Displays the entire store for diagnostic purposes

c) Use the pseudocode from W&W:

- p483 for finding an item in a tree
- p485 for adding an item to a tree

Populate your tree with the data from here:

https://isaaccomputerscience.org/concepts/dsa_datastruct_tree_implementation?examBoard=all&stage=all

so that you can visualise it and test your program.