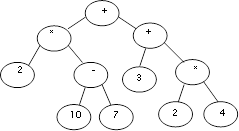
1. The following Binary tree holds the mathematical expression, 
   1. Traverse the tree using inorder, preorder, postorder.
   2. Manually evaluate the numeric value of the expression held by the tree.
   3. Write down a recursive function to evaluate the whole tree or subtree of it. Note::10 is 8 for simplicity in answering
2. Using a BST and an unsorted array of integers, develop an algorithm to sort the array.
3. Create a BST and add to it the following numbers 12, 5, 7, 9, 13, 100, 80, 16, 18, 55, 10, 11, Show the tree after deleting nodes 10, 13, 100. Do you think a different sequence of numbers would be used to have a more compressed tree (same number of elements with minimum depth)
4. Write down a function that returns the maximum and minimum values within a BST.
5. Write a method that counts the number of [leaf] nodes in a binary tree.
6. Write down a program to translate a tree into XML text file. each tree node has two strings one is the tag name and the other is the tag value, each node has also a number of strings pointing to its children. Traversing the XML file tree will be translated into text a file as follows: <tag> value </tag>. When a node has children the XML appears as follows:

<parenttag> parentvalue <childtag> child 1 </childtag><childtag> child 2 </childtag> </parenttag>

The tree could be as following figure:

TagName: Book

TagValue: book11



TagName: Title

TagValue: Data Analysis

Tagname: author

TagValue: Adam



TagName: Unit

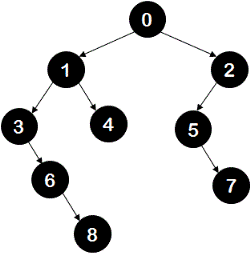
TagValue: Data Gathering

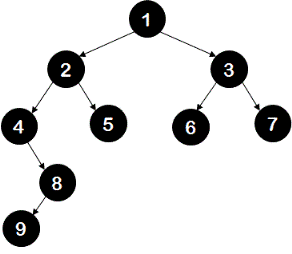
TagName: Unit

TagValue: Pattern mining

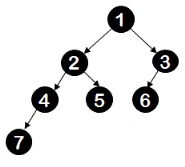
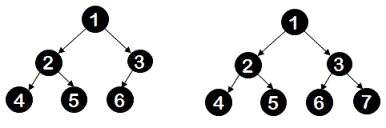
TagName: Unit

TagValue: Data Cleaning

1. Given a binary tree, write a program to print all nodes of that tree which do not have sibling nodes. For example in the following tree, nodes 6, 8, 5, 7 are such nodes because for all of these nodes parent node has only one child.
2. Given a binary tree, print all the root to leaf paths of the tree.
3. Write a program to find the sum of all left leaves of a given binary tree. For example, for the following shown tree output of the program should be 15 as there are two left leaves - node 9 and node 6.



1. Given two trees, find if they are identical. Two trees are said to be identical if they are structurally same and have same data in all nodes.
2. Given two values k1 and k2 (where k1 < k2) and a root pointer to a Binary Search Tree. Print all the keys of tree in range k1 to k2 in order.
3. Write a program to check if a given binary tree is complete tree or not, A binary tree is a complete binary tree if all levels of the tree starting from root node level are filled. Only the last level of the tree is allowed to have an incompletely filled state. Also for tree to be a complete binary tree, all nodes should be placed as left as possible.



1. What is the function fn doing?

|  |  |  |
| --- | --- | --- |
| void left(node \* root){  if(root == NULL) return;  if(root->left != NULL)  left(root->left);  cout<<root->data<<" "; } | void right(node \* root){  if(root == NULL) return;  cout<<root->data<<" ";  if(root->right != NULL)  right(root->right); } | void fn(node \* root) {  left(root);  right(root->right);  } |

1. Describe how can you rebuild a binary tree given Inorder and Preorder traversals and build a tree using the following traversals:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Preorder Traversal: | 1 | 2 | 4 | 8 | 9 | 10 | 11 | 5 | 3 | 6 | 7 |
| Inorder Traversal: | 8 | 4 | 10 | 9 | 11 | 2 | 5 | 1 | 6 | 3 | 7 |

1. The given array shows students' details sorted according to an integer showing the student's result in one course. It is required to create another unique index method that uses a BST which sorts the students according to the surname. Each cell in the BST holds data for the surname and the index of student within the array (pointing at the whole record of data).

* Write down a method to create such a BST starting from the given array
* Write down a find method using surname that finds where a student with surname is found within the array and compute its O()
* Compare this find with the normal linear search using the original array.

