Home Work #3

Sayem Chowdhury

Fr9838

#---------------------------------------------------------- ---------------------------

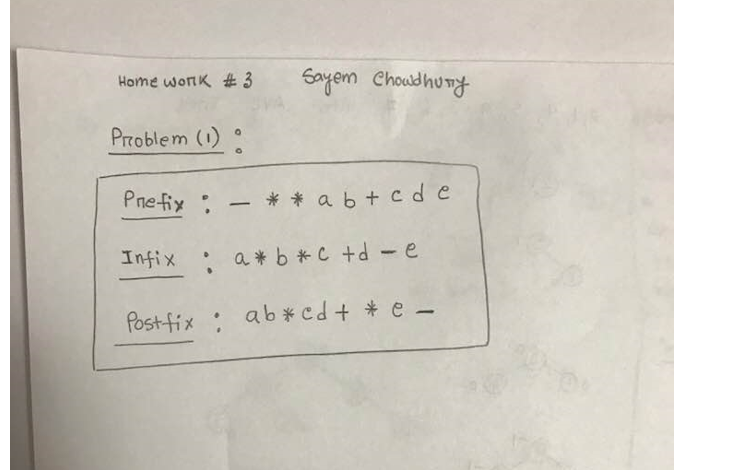
//Sayem Chowdhury

//Home Work #3

//--------------------------------------------------------------

// Problem (1)

//-----------------------------



//problem (2)

//remove the leaf nodes from a given tree and return a pointer to the root of the modified tree

//---------------------------------------------------------------------------------------

Node\* removeAllLeaves(Node\* root)

{

if (root == NULL)

{

return NULL;

}

else

{

if (root->left == NULL && root->right == NULL)

{

delete root;

root = NULL;

}

else

{

root->left = removeAllLeaves(p->left);

root->right = removeAllLeaves(p->right);

}

}

return root; //

}

// problem (3).(a)

// test whether a given binary tree is a binary search tree and return true if it is, otherwise false

//-----------------------------------------------------------------------------------------------------------

bool isSearchTree(Node\* node)

{

if (node == NULL)

return true;

if (node->left != NULL && node->data < node->left->data)

return false;

if (node->right != NULL && node->right->data < node->data)

return false;

return isSearchTree(node->left) && isSearchTree(node->right);

}

//========================================================================

// problem 3.(b)

//------------------------------------------------------------------------------------------

//find a path from root node to a given node and store the found path in a vector path

// return true if a path exists otherwise false

bool findPath(Node \*root, std::vector<int> &path, int target)

{

if (!root)// if root=NULL tree is empty, so there is no path exists

return false;

path.push\_back(root->data); // push the root data into the vector path

if (root->data == target) // when we found the target node

return true;

//check left sub tree

if (findPath(root->left, path, target))

return true;

//check right sub tree

else if (findPath(root->right, path, target))

return true;

//remove item already pushed, if path is not found for target item

path.pop\_back();

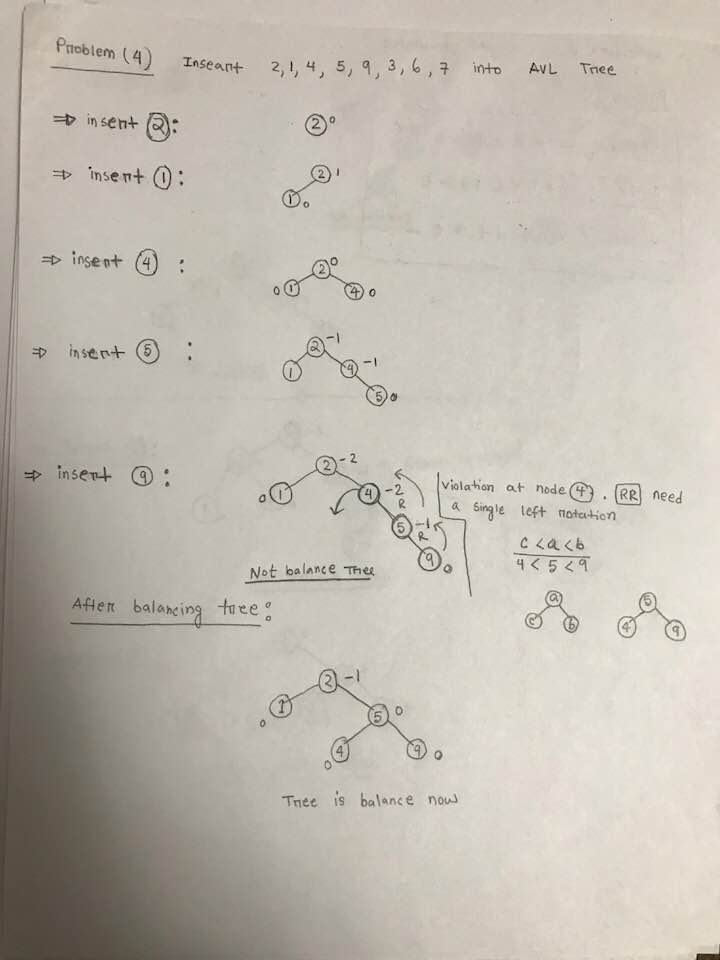
return false;

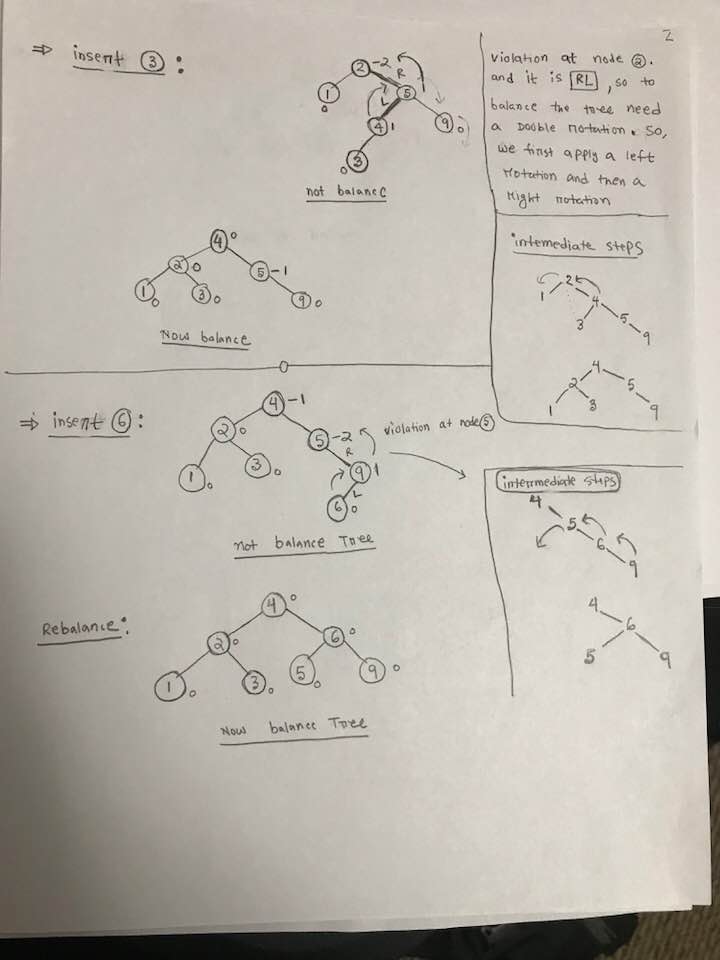
}

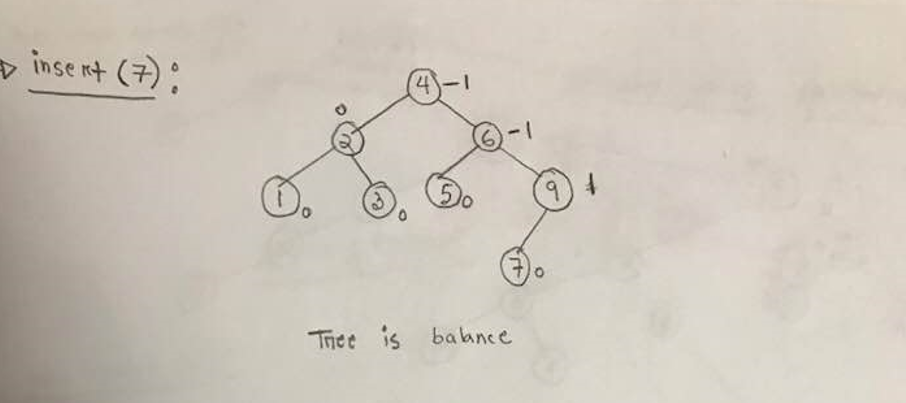
#-----------------------------------------------------------------------

//problem #4

//-------------------------------------







//Problem #5

#-----------------------------------------------

