Muhammad Abdullah

L1f21bscs0379

DSA\_Assignment\_03

D12

Task 1

Solution:

The in-order traversal of the given tree will visits the nodes in ascending order. Starting from the leftmost node it traverses the left subtree visits the current node and then moves to the right subtree.

The in-order traversal of the tree is: 5, 4, 7, 9, 10, 18, 15, 21, 22, 23, 25, 20, 35, 30, 45, 26, 60, 40, 50.

The pre-order traversal explores the nodes in a top-down manner. It starts from the root and visits the current node before traversing the left and right branches so the pre-order traversal of the given tree is: 30, 20, 15, 5, 4, 9, 7, 10, 18, 25, 22, 26, 21, 23, 40, 35, 45, 60, 50.

The post-order traversal visits the nodes in a bottom-up fashion. It first explores the left and right subtrees before visiting the current node…

It will be: 4, 7, 10, 9, 5, 18, 21, 23, 22, 26, 25, 15, 35, 60, 45, 50, 40, 20, 30.

Muhammad Abdullah

L1f21bscs0379

Note: I wrote all code in visual studio code as I got some issues while compiling code in visual studio 2013.

Task 3

Solution:

#include<iostream>

using namespace std;

/\*

Muhammad Abdullah

L1F21BSCS0379

D12

DSA Assignment 3

\*/

class RibbonCutter

 {

public:

    int getMaxPieces(int n, int arr[], int size)

     {

        if (n == 0)

        {

            return 0;

        }

Muhammad Abdullah

L1f21bscs0379

        if (n < 0 || size == 0)

         {

            return -1;

        }

        int maxPieces = -1;

        for (int i = 0; i < size; i++)

        {

            int remainingLength = n - arr[i];

            int pieces = getMaxPieces(remainingLength, arr, size);

            if (pieces != -1)

             {

                maxPieces = std::max(maxPieces, pieces + 1);

            }

        }

        return maxPieces;

    }

};

int main()

 {

    int ribbonLength = 5;

    int pieces[] = {1, 2, 3};

    int size = sizeof(pieces) / sizeof(pieces[0]);

    RibbonCutter cutter;

    int maxPieces = cutter.getMaxPieces(ribbonLength, pieces, size);

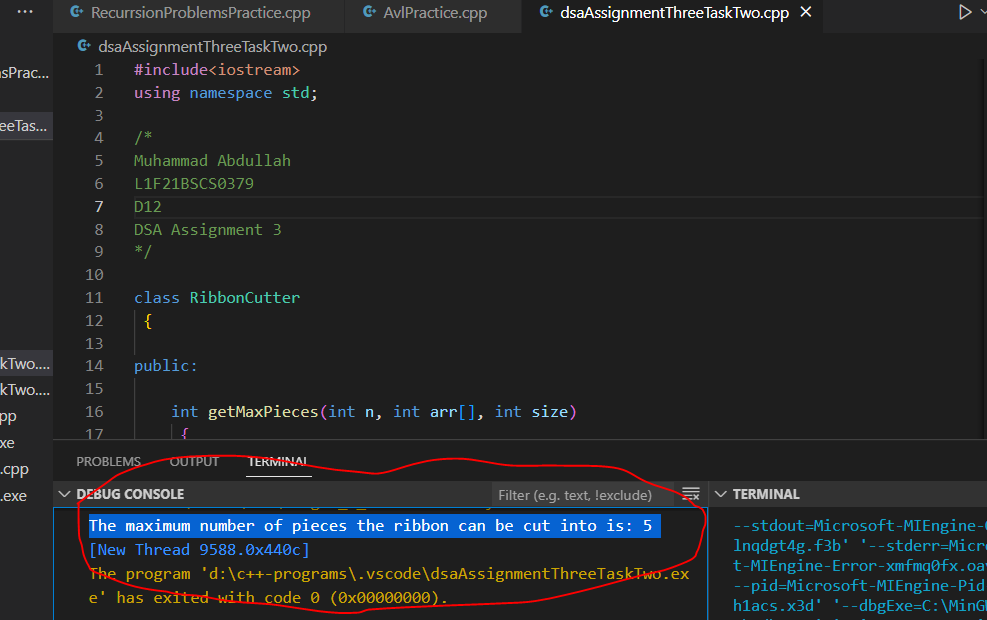
std::cout << "The maximum number of pieces the ribbon can be cut into is: " << maxPieces << std::endl;

    return 0;

}

Muhammad Abdullah

L1f21bscs0379



Task 3

Solution:

#include <iostream>

using namespace std;

/\*

Muhammad Abdullah

L1F21BSCS0379

D12

DSA Assignment 3

\*/

class TreeNode

Muhammad Abdullah

L1f21bscs0379

public:

    int val;

    TreeNode\* left;

    TreeNode\* right;

    TreeNode(int value)

     {

        val = value;

        left = nullptr;

        right = nullptr;

    }

};

class BinaryTree

 {

public:

    int diameter(TreeNode\* root)

    {

        if (root == nullptr)

        {

            return 0;

        }

        int leftHeight = height(root->left);

        int rightHeight = height(root->right);

        int leftDiameter = diameter(root->left);

        int rightDiameter = diameter(root->right);

        int currentDiameter = leftHeight + rightHeight;

        return std::max(currentDiameter, std::max(leftDiameter, rightDiameter));

    }

private:

    int height(TreeNode\* node)

     {

        if (node == nullptr)

         {

Muhammad Abdullah

L1f21bscs0379

    return 0;

        }

        int leftHeight = height(node->left);

        int rightHeight = height(node->right);

        return 1 + std::max(leftHeight, rightHeight);

    }

};

int main()

{

    TreeNode\* root = new TreeNode(1);

    root->left = new TreeNode(4);

      root->left->right = new TreeNode(5);

          root->left->left = new TreeNode(6);

 root->right = new TreeNode(3);

    root->right->right = new TreeNode(8);

        root->right->right->left = new TreeNode(7);

    root->right->right->left->right = new TreeNode(9);

    BinaryTree binaryTree;

    int treeDiameter = binaryTree.diameter(root);

    std::cout << "Diameter of the binary tree: " << treeDiameter << std::endl;

    return 0;

}

Muhammad Abdullah

L1f21bscs0379

