**Taaha Hussain Khan**

**L1F21BSCS0917**

**D12**

**Assignment 03**

**Task 01**

In-order:

*Left-Root-Right*

**4,5,7,9,10,15,18,20,21,22,23,25,26,30,35,40,45,50,60**

Pre-order:

*Root-Left-Right*

**30,20,15,5,4,9,7,10,18,25,22,21,23,26,40,35,50,45,60**

Post-order:

*Left-Right-Root*

**4,7,10,9,5,18,15,21,23,22,26,25,20,35,45,60,50,40,30**

**Task 02**

#include <iostream>

#include <math.h>

using namespace std;

void displayPieces(int pieces[], int currentPieces[], int currentLength, int target, int n);

int maxPieces(int pieces[], int length, int n);

int main()

{

    int n = 5;

    int pieces[] = {1, 2, 3};

    int currentPieces[100] = {0};

    cout << "The ribbon can be cut into pieces of length: ";

    cout << endl;

    displayPieces(pieces, currentPieces, 0, n, n);

    cout << endl;

    cout << "The maximum pieces in which a ribbon can cut into is " << n << " , so the answer is  " << maxPieces(pieces, 3, n) << endl;

}

int maxPieces(int pieces[], int length, int n)

{

    int countMax = -1;

    if (n == 0)

    {

        return 0;

    }

    for (int i = 0; i < length; i++)

    {

        if (n >= pieces[i])

        {

            int count = maxPieces(pieces, length, n - pieces[i]);

            if (count != -1)

            {

                countMax = max(countMax, count + 1);

            }

        }

    }

    int cm = countMax;

    return cm;

}

void displayPieces(int pieces[], int currentPieces[], int currentLength, int t, int n)

{

    if (t == 0)

    {

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

        {

            cout << currentPieces[i] << " ";

            if (i != currentLength - 1)

               {

                 cout << " + ";

            }

        }

        cout << " = " << n << endl;

        return;

    }

    for (int i = 0; i < 3; i++)

    {

        if (t >= pieces[i])

        {

            currentPieces[currentLength] = pieces[i];

            displayPieces(pieces, currentPieces, currentLength + 1, t - pieces[i], n);

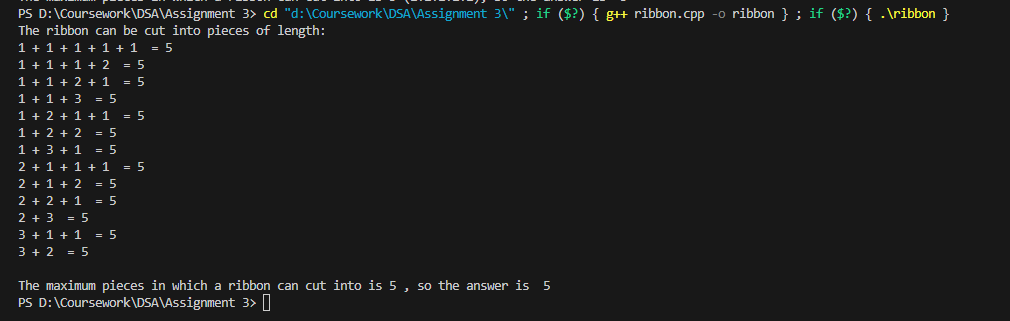
            currentPieces[currentLength] = 0;

        }

    }

}

**Output:**



**Task 03**

#include <iostream>

using namespace std;

class node

{

public:

    int data;

    node \*left;

    node \*right;

    node(int data)

    {

        this->data = data;

        left = NULL;

        right = NULL;

    }

};

int diameter(node \*root, int \*height);

int main() {

    node\* root = new node(1);

    root->left = new node(4);

    root->left->left = new node(2);

    root->left->right = new node(5);

    root->left->right->left = new node(6);

    root->right = new node(3);

    root->right->right = new node(8);

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

    root->right->right->left->left = new node(0);

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

    int h = 0;

    cout << "Diameter of the tree is: ";

    cout << diameter(root, &h);

    return 0;

}

int diameter(node \*root, int \*height)

{

    if (root == NULL)

    {

        \*height = 0;

        return 0;

    }

    int left\_height = 0, right\_height = 0;

    int left\_diameter = diameter(root->left, &left\_height);

    int right\_diameter = diameter(root->right, &right\_height);

    int curr\_diameter = left\_height + right\_height + 1;

    \*height = max(left\_height, right\_height) + 1;

    return max(curr\_diameter, max(left\_diameter, right\_diameter));

}

**Output:**

