**03.11.2020-PRACTICE SESSION CODES**

**UNIQUE BST’S:**

#include <bits/stdc++.h>

using namespace std;

int main()

{

    int t;

    cin >> t;

    while (t--)

    {

        int n;

        float pos = 1;

        cin >> n;

        for (int i = 2; i <= n; i++)

        {

            float temp = (2 \* i - 1);

            temp \*= 2;

            temp /= i + 1;

            pos \*= temp;

        }

        cout << pos << endl;

    }

    return 0;

}

**CHECK INORDER OF BST:**

#include <bits/stdc++.h>

using namespace std;

int main()

{

    int t;

    cin >> t;

    while (t--)

    {

        int n, val;

        bool flag = true;

        cin >> n;

        cin >> val;

        n--;

        while (n--)

        {

            int temp;

            cin >> temp;

            if (temp <= val)

                flag = false;

            val = temp;

        }

        cout << (flag ? 1 : 0) << endl;

    }

    return 0;

}

**SHORTEST DISTANCE BETWEEN TWO NODES IN AN INFINITE BST**

##include <bits/stdc++.h>

using namespace std;

int main()

{

    int t;

    cin >> t;

    while (t--)

    {

        int x, y;

        cin >> x >> y;

        int cnt = 0;

        while (x != y && x && y)

        {

            int \*a = (x > y) ? &x : &y;

            \*a /= 2;

            cnt++;

        }

        cout << cnt << endl;

    }

    return 0;

}

**CHECKING THE PREORDER TRAVERSAL OF AN BST:**

#include <bits/stdc++.h>

using namespace std;

int main()

{

    int t;

    cin >> t;

    while (t--)

    {

        int n;

        cin >> n;

        int flag = true;

        vector<int> a(n);

        cin >> a[0] >> a[1];

        for (int i = 2; i < n; i++)

        {

            cin >> a[i];

            if (a[i - 2] < a[i - 1] && a[i] < a[i - 2])

                flag = false;

        }

        cout << (flag ? 1 : 0) << endl;

    }

    return 0;

}

**CLONE A BINARY TREE:**

#include <bits/stdc++.h>

using namespace std;

struct Node

{

    int data;

    struct Node \*left;

    struct Node \*right;

    struct Node \*random;

    Node(int x)

    {

        data = x;

        left = NULL;

        right = NULL;

        random = NULL;

    }

};

Node \*cloneTree(Node \*);

int printInorder(Node \*a, Node \*b)

{

    if ((a == NULL and b == NULL) or (a->random == NULL and b->random == NULL))

        return 1;

    if (a->random->data == b->random->data and printInorder(a->left, b->left) and printInorder(a->right, b->right))

        return 1;

    return false;

}

void inorder(Node \*root)

{

    if (root == NULL)

        return;

    inorder(root->left);

    cout << root->data << " ";

    inorder(root->right);

}

int main()

{

    int t;

    scanf("%d\n", &t);

    while (t--)

    {

        map<int, Node \*> m;

        int n;

        scanf("%d", &n);

        struct Node \*root = NULL;

        struct Node \*child;

        while (n--)

        {

            Node \*parent;

            char lr;

            int n1, n2;

            scanf("%d %d %c", &n1, &n2, &lr);

            if (m.find(n1) == m.end())

            {

                parent = new Node(n1);

                m[n1] = parent;

                if (root == NULL)

                    root = parent;

            }

            else

                parent = m[n1];

            child = new Node(n2);

            if (lr == 'L')

            {

                parent->left = child;

                m[n2] = child;

            }

            else if (lr == 'R')

            {

                parent->right = child;

                m[n2] = child;

            }

            else

            {

                parent->random = m[n2];

            }

        }

        Node \*t = cloneTree(root);

        if (t == root)

            cout << 0 << endl;

        else

            cout << printInorder(root, t);

        cout << endl;

    }

    return 0;

}

Node \*cloneTree(Node \*tree);

Node \*cloneTree(Node \*tree)

{

    if (!tree)

        return NULL;

    Node \*root = new Node(tree->data);

    root->left = cloneTree(root->left);

    root->right = cloneTree(root->right);

    return root;

}

**TERNARY OPERATION TO BINARY TREE:**

#include <bits/stdc++.h>

using namespace std;

struct Node

{

    char data;

    struct Node \*left;

    struct Node \*right;

    Node(char x)

    {

        data = x;

        left = NULL;

        right = NULL;

    }

};

void preorder(Node \*root)

{

    if (root == NULL)

        return;

    cout << root->data << " ";

    preorder(root->left);

    preorder(root->right);

}

Node \*convertExpression(string str, int i);

int main()

{

    int t;

    cin >> t;

    while (t--)

    {

        string str;

        cin >> str;

        Node \*root = convertExpression(str, 0);

        preorder(root);

        cout << endl;

    }

}

Node \*convertExpression(string str, int i)

{

    if (i > str.length())

        return NULL;

    Node \*root = new Node(str[i++]);

    if (i < str.length() && str[i] == '?')

        root->left = convertExpression(str, i + 1);

    if (i < str.length() && str[i] == ':')

        root->right = convertExpression(str, i + 1);

    return root;

}