**A Practical Activity Report For**

**Data Structures and Algorithms (UCS406)**

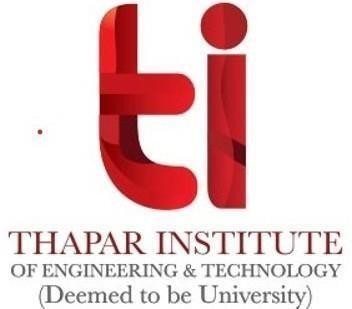
Submitted By: **Vivek Arora**

**101715178**

**(ENC 8)**

Submitted To:

**Dr. Sanjay Sharma**



**ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT THAPAR INSTITUTE OF ENGINEERING &TECHNOLOGY, (DEEMED TO BEUNIVERSITY),**

**PATIALA, PUNJAB**

**ASSIGNMENT-8**

**QUESTION 1** Write a program for creating a binary search tree (BST) from a given array of elements.

#include<iostream>

using namespace std;

class BinarySearchTree{

public:

int size;

int\* array;

void insertElement(int x);

void searchElement(int x);

void inOrder(int currentIndex);

void preOrder(int currentIndex);

void postOrder(int currentIndex);

void parent(int x);

int extendSize(int x);

BinarySearchTree (int size) {

     this -> size = extendSize(size);

     //cout << this -> size << endl;

     this -> array = new int[this -> size];

     for(int x = 0; x < this -> size; x++){

         array[x] = NULL;

     }

}

};

int BinarySearchTree::extendSize(int x) {

int value = 0;

for(int y = 0; y < x + 1; y++) {

     value = (2 \* value) + 2;

}

return value;

}

void BinarySearchTree::insertElement(int x) {

int currentIndex = 0;

cout << "Adding: " << x;

while(true) {

     if(array[currentIndex] == NULL){

         array[currentIndex] = x;

         cout << " Inserted at index: " << currentIndex << endl;

         break;

     }else if(array[currentIndex] <= x) {

         if(array[currentIndex] == x){

             cout << "ERROR!-- Repeating element" << endl;

             break;

         }else

         cout << " Right ";

         currentIndex = (2 \* currentIndex + 2);

     }else if(array[currentIndex] >= x) {

          if(array[currentIndex] == x){

             cout << "ERROR!-- Repeating element" << endl;

             break;

         }else

         cout << " Left ";

         currentIndex = (2 \* currentIndex + 1);

     }

}

}

void BinarySearchTree::searchElement(int x){

int currentIndex = 0;

while (true) {

         if (array[currentIndex] == NULL) {

         cout << "Not Found" << endl;

         break;

         }

         if (array[currentIndex] == x) {

         cout << "Found at index: " << currentIndex << endl;

         break;

         }

         else if(array[currentIndex] < x) {

         currentIndex = (2 \* currentIndex + 2);

     }

         else if(array[currentIndex] > x) {

         currentIndex = (2 \* currentIndex + 1);

     }

}

}

void BinarySearchTree::parent(int x){

while (x != 0) {

     x = (x-1) / 2;

     cout << "---";

}

}

void BinarySearchTree::inOrder(int currentIndex){

if(array[currentIndex] != NULL) {

         inOrder(2 \* currentIndex + 1);

         parent(currentIndex);

         cout << array[currentIndex] << endl;

         inOrder(2 \* currentIndex + 2);

}

}

void BinarySearchTree::postOrder(int currentIndex) {

if(array[currentIndex] != NULL){

     postOrder(2 \* currentIndex + 1);

     postOrder(2 \* currentIndex + 2);

     parent(currentIndex);

     cout << array[currentIndex] << " " << endl;

}

}

void BinarySearchTree::preOrder(int currentIndex) {

if(array[currentIndex] != NULL) {

     preOrder(2 \* currentIndex + 1);

     parent(currentIndex);

     cout << array[currentIndex] << " " << endl;

     preOrder(2 \* currentIndex + 2);

}

}

int main () {

BinarySearchTree tree(5);

tree.insertElement(4);

tree.insertElement(6);

tree.insertElement(9);

tree.insertElement(3);

tree.insertElement(2);

tree.searchElement(1);

tree.inOrder(0);

};

**QUESTION 2 Write a program to insert an element in a BST.**

#include<iostream>

using namespace std;

class Node

{

    int data;

    Node\* left,\*right;

public:

    Node(int data)

    {    this->data=data;

    left=right=NULL;

    }

    Node\* newNode(int d)

    {    Node\* root=new Node(d);

    return root;

    }

    Node\* Insert(Node\* root, int x)

    {    if(root==NULL)

    {

    root=newNode(x);

    return root;

    }

    if(x<=root->data)

    root->left=Insert(root->left,x);

    else if(x>root->data)

    root->right=Insert(root->right,x);

    }

    void preorder(Node\* root)

    {    if(root==NULL)

    return;

    cout<<endl<<root->data<<" ";

    preorder(root->left);

    preorder(root->right);

    }

    void postorder(Node\* root)

    {    if(root==NULL)

    return;

    postorder(root->left);

    postorder(root->right);

    cout<<endl<<root->data<<" ";

    }

    void inorder(Node\* root)

    {    if(root==NULL)

    return;

    inorder(root->left);

    cout<<endl<<root->data<<" ";

    inorder(root->right);

    }

};

int main()

{

    Node\* root=NULL;

    int n,no;

    cout<<"\n How many elements(nodes) you want to enter in the tree\n";

    cin>>n;

    for(int i=0;i<n;i++)

    {    cin>>no;

    root=root->Insert(root,no);

    }

    cout<<"\n Preorder: ";

    root->preorder(root);

    cout<<"\n Postorder: ";

    root->postorder(root);

    cout<<"\n Inorder: ";

    root->inorder(root);

return 0;

}

**QUESTION 3 Write a recursive as well as iterative program for search in a BST.**

**RECURSIVE --**

#include<iostream>

using namespace std;

class Node

{

    int data;

    Node \*left,\*right;

public:

    Node(int data)

    {    this->data=data;

    left=right=NULL;

    }

    Node\* newNode(int data)

    {    Node\* root2=new Node(data);

    return root2;

    }

    Node\* insert(Node\* root,int x)

    {    if(root==NULL)

    {    root=newNode(x);

    return root;

    }

    if(x<=root->data)

    root->left=insert(root->left,x);

    else root->right=insert(root->right,x);

    }

    bool search(int x)

    {    if(x==this->data)

    return true;

    else if(x<=this->data)

    this->left->search(x);

    else this->right->search(x);

    }

};

int main()

{

    Node\*root=NULL;

    int n,no,x;

    cout<<"\nHow many elements/nodes?\n";

    cin>>n;

    for(int i=0;i<n;i++)

    {    cin>>no;

    root=root->insert(root,no);

    }

    cout<<"\nEnter the no. to be searched ";

    cin>>x;

    bool temp=root->search(x);

    if(temp==true) cout<<"\nFound\n";

    else cout<<"\nNot Found\n";

return 0;

}

**QUESTION Write a program for performing to print the elements of a BST after performing In-order**

**Traversal.**

#include<iostream>

using namespace std;

class Node

{    int data;

    Node\* left;

    Node\* right;

    public:

    Node(int data)

    {    this->data=data;

    this->left=NULL;

    this->right=NULL;

    }

    Node\* push(Node\* root)

    {

    root=new Node(1);

    root->left=new Node(2);

    root->left->left=new Node(3);

    root->left->right=new Node(4);

    root->right=new Node(5);

    root->right->left=new Node(6);

    root->right->right=new Node(7);

    return root;

    }void inorder(Node\* root)

    {    if(root==NULL)

    return;

    inorder(root->left);

    cout<<root->data<<" ";

    inorder(root->right);

    }

    void preorder(Node\* root)

    {    if(root==NULL)

    return;

    cout<<root->data<<" ";

    preorder(root->left);

    preorder(root->right);

    }

    void postorder(Node\* root)

    {    if(root==NULL)

    return;

    postorder(root->left);

    postorder(root->right);

    cout<<root->data<<" ";

    }

};

int main()

{    Node\* root;

    root=root->push(root);

    cout<<"\nInorder: ";

    root->inorder(root);

    cout<<"\nPreorder: ";

    root->preorder(root);

    cout<<"\nPostorder: ";

    root->postorder(root);

    cout<<endl;

return 0;

}

**QUESTION 5 Write a program to bubble sort a given array of elements.**

#include<iostream>

using namespace std;

void bubble\_sort(int A[],int n){

    for(int j=0;j<n-1;j++){

    int flag=0;

    for(int i=0;i<n-j-1;i++){

    if(A[i+1]<A[i]){

    flag=1;

    int temp=A[i];

    A[i]=A[i+1];

    A[i+1]=temp;

    }

    }

    if(flag==0)

    break;

    }

}

int main(){

    int arr[]={2,7,4,1,5,3,6,9,10,57};

n= \*(&arr + 1) - arr;

    bubble\_sort(arr,n);

    cout<<endl;

    for(int i=0;i<n;i++){

    cout<<arr[i]<<" ";

    }

    cout<<endl;

}