For this project assignment, you will solve a problem based on what you have learnt in this course.

**Instructions**

* Write your name and SNU\_ID in the header of this document.
* Assignment submitted after the due date will not be evaluated and a score of zero will be awarded.
* Plagiarized assignments will also be awarded zero.
* Upload a word version of this document.
* Properly document/comment your code, followed by snapshots of output as desired.

**Submitting this Assignment**

* You will submit (upload) this assignment in Blackboard.
* Email/paper submissions will not be accepted.
* Name this document as Project\_CSD203-2020\_John\_Bill.doc in case the first names of group members are John and Bill respectively.

**Grading Criteria**

**This assignment has 13 points (with weightage of 13% in your overall 100 points). Points will be awarded as follows:**

1. Functionality – **10 points**
2. Look and Feel of node creation, deletion and searching implementations – **3 points**

**Project Problem**

Write a java program to create a **Binary Search Tree** that will make use of several JavaFX components, event handling, graphics, and Java Collections Framework to implement. GUI must contain buttons to perform following operations:

1. Insert - to insert a node (element) into the tree (3 Marks)
2. Delete - to delete a node from the tree (3 Marks)
3. Find- to search an element in the tree (1 Marks)
4. Print – to print the sorted list of elements (1 Marks)

Program should keep updating the following details at the bottom of the Frame:

1. height of the tree (1 Marks)
2. number of vertices (1 Marks)

**Filename: Assignment3.java**

package assignment3;

import java.io.IOException;

import static javafx.application.Application.launch;

import javafx.application.Application;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.\*;

import javafx.stage.Stage;

/\*\*

\*

\* @author sam

\*/

public class Assignment3 extends Application

{

public void start(Stage primaryStage) throws IOException

{

// Create a pane

Parent root = FXMLLoader.load(getClass().getResource("FXML.fxml"));

Scene scene = new Scene(root, 300, 275);

primaryStage.setTitle("BST GUI (JAVAFX Assignment 3)");

primaryStage.setScene(scene);

primaryStage.show();

}

public static void main(String[] args)

{

launch(args);

}

}

**Filename: FXMLController.java**

package assignment3;

import static java.lang.Integer.parseInt;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.scene.control.TextField;

import javafx.fxml.Initializable;

import javafx.scene.control.\*;

import javafx.scene.layout.BorderPane;

import java.net.URL;

import java.util.ArrayList;

import java.util.ResourceBundle;

import javafx.geometry.Point2D;

import javafx.scene.canvas.\*;

import javafx.scene.paint.Color;

/\*\*

\* FXML Controller class

\*

\* @author sam

\*/

class GTree extends Canvas

{

public Tree tree;

public String listTraversed;

public int maxHeightAllowed;

public Color defaultColor;

GTree()

{

tree=new Tree();

listTraversed="";

maxHeightAllowed=6;

defaultColor=Color.LIGHTGREEN;

widthProperty().addListener(evt -> draw()); //for listening to change in width

heightProperty().addListener(evt -> draw()); //for listening to change in height

}

//render elements of the tree on GUI

void draw()

{

GraphicsContext gc = getGraphicsContext2D();

gc.clearRect(0, 0, this.getWidth(), this.getHeight());

// If the tree is not empty; draw the lines and circles

if (tree.root != null)

{

int ht = tree.height;

double diameter, xMin=100, xMax, yMin=0, yMax;

int divBy = maxHeightAllowed + 3;

yMax= this.getHeight();

xMax= this.getWidth()-100;

diameter= yMax / divBy;

Point2D center= new Point2D(((xMin + xMax) / 2), (yMin + diameter/2));//skipping first division

drawTree(gc, tree.root, xMin, xMax, center, diameter);

}

}

void drawTree(GraphicsContext gc, Node treeNode, double xMin, double xMax, Point2D center, double diameter)

{

// Set the stroke Color

gc.setStroke(Color.BROWN);

if(treeNode!=null)

{

// Set fill color

gc.setFill(treeNode.color);

gc.fillOval(center.getX()-diameter/4,center.getY()-diameter/4, diameter/2, diameter/2);

treeNode.color = defaultColor;

gc.strokeText(""+treeNode.key, center.getX()-6,center.getY()+5);

Point2D center2;

if (treeNode.left != null)

{

// Determine the start and end points of the line

center2 = new Point2D((xMin+center.getX())/2,(center.getY()+diameter));

gc.strokeLine(center.getX()-diameter/5,center.getY()+diameter/8,center2.getX(),center2.getY());

// Recurse left circle nodes

drawTree(gc, treeNode.left, xMin, center.getX(), center2, diameter);

}

// If right node is not null then draw a line to it

if (treeNode.right != null)

{

// Determine the start and end points of the line

center2 = new Point2D((center.getX()+xMax)/2,(center.getY()+diameter));

gc.strokeLine(center.getX()+diameter/5,center.getY()+diameter/8,center2.getX(),center2.getY());

// Recurse left circle nodes

drawTree(gc, treeNode.right, center.getX(), xMax, center2, diameter);

}

}

}

void storeList(ArrayList<Integer> l, char rev)

{

//convert array list to string

listTraversed = "";

if (l.isEmpty())

listTraversed=" Empty tree";

else {

if(rev == 'r')

{

for(int i = l.size()-1; i>=0; i--)

{

listTraversed += l.get(i).toString() +" ";

}

}

else

{

for (Integer a:l)

{

listTraversed += a.toString() +" ";

}

}

}

}

void insert(int element)

{

tree.insert(element);

draw();

}

//searching an element

String search(int element)

{

String str;

boolean b=tree.search(element);

draw();

str=element+"";

if (b==false)

str = str + " Not found";

else str=str+" Found";

return str;

}

//deleting an element

String delete(int element)

{

String str;

boolean b=tree.delete(element);

// node not found case to be handled

draw();

str=element+"";

if (b==false)

str = str + " Not found";

else str = str + " Deleted";

return str;

}

//preorder traversal

void preorder()

{

ArrayList<Integer> l=new ArrayList<>();

l= tree.preorder(l,tree.root);

storeList(l,' ');

draw();

}

//postorder traversal

void postorder()

{

ArrayList<Integer> l=new ArrayList<>();

l= tree.postorder(l,tree.root);

storeList(l,' ');

draw();

}

//inorder traversal

void inorder(char rev)

{

ArrayList<Integer> l=new ArrayList<>();

l= tree.inorder(l,tree.root);

storeList(l,rev);

draw();

}

//clear the tree

void clear()

{

tree.clear();//should empty the tree

draw();

}

}

public class FXMLController implements Initializable{

// Panels and other GUI components

@FXML private BorderPane rootPane;

@FXML private TextArea valueList;

@FXML private TextArea counters;

@FXML private TextField value;

private GTree gtree;

/\*\*

\* Constructs the GUI components and performs events for displaying and

\* changing the data in the binary tree.

\*/

/\*Initializable and the method it adds are used when you want to interact with stuff injected with @FXML.

During construction those variables aren't filled so you cannot interact with them so JavaFX will call

the Initializable interface after everything is set up. At that point those variables are available and

can be manipulated.\*/

@Override

public void initialize(URL location, ResourceBundle resources)

{

// The center panel is for drawing the tree

gtree = new GTree();

rootPane.setCenter(gtree);

// Bind canvas size to stack pane size.

gtree.widthProperty().bind(rootPane.widthProperty());

gtree.heightProperty().bind(rootPane.heightProperty().subtract(50));

gtree.clear();

valueList.setText("");

counters.setText("Height: "+gtree.tree.height + " No. of Nodes: "+gtree.tree.noOfNodes);

}

@FXML protected void insertButton(ActionEvent event)

{

String strV=value.getText();

if (strV.isBlank())

{

counters.setText("Height: "+gtree.tree.height + " No. of Nodes: "+gtree.tree.noOfNodes+

" No key value given");

}

else

{

gtree.insert(parseInt(strV));

valueList.setText(valueList.getText()+" "+strV + "i");

counters.setText("Height: "+gtree.tree.height + " No. of Nodes: "+gtree.tree.noOfNodes);

}

}

@FXML protected void searchButton(ActionEvent event) {

String strV=value.getText();

if (strV.isBlank())

{

counters.setText("Height: "+gtree.tree.height + " No. of Nodes: "+gtree.tree.noOfNodes+

" No key value given");

}

else

{

String str = gtree.search(parseInt(strV));

valueList.setText(valueList.getText()+" "+strV + "s");

counters.setText("Height: "+gtree.tree.height + " No. of Nodes: "+gtree.tree.noOfNodes+

" "+str);

}

}

@FXML protected void deleteButton(ActionEvent event)

{

String strV=value.getText();

if (strV.isBlank())

{

counters.setText("Height: "+gtree.tree.height + " No. of Nodes: "+gtree.tree.noOfNodes+

" No key value given");

}

else

{

String str=gtree.delete(parseInt(strV));

valueList.setText(valueList.getText()+" "+strV + "d");

counters.setText("Height: "+gtree.tree.height + " No. of Nodes: "+gtree.tree.noOfNodes+

" "+str);

}

}

@FXML protected void clearButton(ActionEvent event)

{

gtree.clear();

valueList.setText("");

counters.setText("Height: "+gtree.tree.height + " No. of Nodes: "+gtree.tree.noOfNodes);

}

@FXML protected void inorderButton(ActionEvent event)

{

gtree.inorder(' ');

counters.setText("Height: "+gtree.tree.height + " No. of Nodes: "+gtree.tree.noOfNodes+" "+"Inorder Traversal: "+

gtree.listTraversed);

}

@FXML protected void preorderButton(ActionEvent event)

{

gtree.preorder();

counters.setText("Height: "+gtree.tree.height + " No. of Nodes: "+gtree.tree.noOfNodes+" "+"Preorder Traversal: "+

gtree.listTraversed);

}

@FXML protected void postorderButton(ActionEvent event)

{

gtree.postorder();

counters.setText("Height: "+gtree.tree.height + " No. of Nodes: "+gtree.tree.noOfNodes+" "+"Postorder Traversal: "+

gtree.listTraversed);

}

@FXML protected void sortASCButton(ActionEvent event)

{

gtree.inorder(' ');

counters.setText("Height: "+gtree.tree.height + " No. of Nodes: "+gtree.tree.noOfNodes+" "+"Sorted List Ascending: "+

gtree.listTraversed);

}

@FXML protected void sortDESCButton(ActionEvent event)

{

gtree.inorder('r');

counters.setText("Height: "+gtree.tree.height + " No. of Nodes: "+gtree.tree.noOfNodes+" "+"Sorted List Descending: "+

gtree.listTraversed);

}

}

**Filename: Tree.java**

package assignment3;

import java.util.ArrayList;

import javafx.scene.paint.Color;

/\*\*

\*

\* @author sam

\*/

//implements BST functionality

class Node

{

public int key;

public Node left;

public Node right;

public Color color;

Node(int key,Color clr)

{

this.key=key;

this.left=null;

this.right=null;

color=clr;

}

Node(int key)

{

this.key=key;

this.left=null;

this.right=null;

color=Color.LIGHTGREEN;

}

Node()

{

this.left=null;

this.right=null;

}

}

public class Tree

{

public Node root;

public int height;

public int noOfNodes;

//default constructor

Tree()

{

root=null;

height=0;

noOfNodes=0;

}

//insert an element in BST

Node insertTree(Node n, int key)

{

if(n==null)

{

Node leaf=new Node(key,Color.CORAL);

n=leaf;

return n;

}

else

{

if (key <= n.key)

n.left = insertTree(n.left, key);

else if (key > n.key)

n.right = insertTree(n.right, key);

/\* return the (unchanged) node pointer \*/

return n;

}

}

int height()

{

return (treeHeight(root));

}

int treeHeight(Node n)

{

if(n == null)

{

return 0;

}

int left=treeHeight(n.left);

int right=treeHeight(n.right);

if(left>right)

{

return (left+1);

}

else

{

return (right+1);

}

}

void insert(int element)

{

root=insertTree(root,element);

noOfNodes++;

height=height();

}

Node searchTree(Node n,int key)

{

if (n==null)

return n;

if(n.key==key)

{

n.color=Color.CYAN;

return n;

}

if (n.key < key)

return searchTree(n.right, key);

return searchTree(n.left, key);

}

//searching an element

boolean search(int element)

{

if(searchTree(root,element)!=null)

return true;

else

return false;

}

//deleting an element

boolean delete(int elem)

{

boolean found[]={false};

root=deleteTree(root,elem,found);

height=height();

return found[0];

}

Node insertDel(Node n)

{

Node parent=n;

Node rt=n.right;

Node child= rt.left;

while (child!=null)

{

parent=rt;

rt=rt.left;

child=rt.left;

}

n.key=rt.key;

parent.left=rt.right;

return n;

}

Node deleteTree(Node n,int elem, boolean [] found)

{

if (n==null)

{

found[0]=false;

n= null;

}

else if(elem< n.key)

{

n.left=deleteTree(n.left,elem, found);

}

else if(elem>n.key)

{

n.right=deleteTree(n.right,elem, found);

}

else if(elem==n.key)

{

found[0]=true;

noOfNodes--;

if(n.left==null && n.right==null)

{

n= null;

}

else if(n.right!=null && n.left!=null)

{

n=insertDel(n);

}

else if(n.right!=null && n.left==null)

{

n=n.right;

}

else if(n.right==null && n.left!=null)

{

n=n.left;

}

}

return n;

}

ArrayList<Integer> inorder(ArrayList<Integer> p,Node n)

{

if (n != null)

{

inorder(p,n.left);

p.add(n.key);

inorder(p,n.right);

}

return p;

}

//postorder traversal

ArrayList<Integer> postorder(ArrayList<Integer> p,Node n)

{

if (n != null)

{

postorder(p,n.left);

postorder(p,n.right);

p.add(n.key);

}

return p;

}

ArrayList<Integer> preorder(ArrayList<Integer> p,Node n)

{

if (n != null)

{

p.add(n.key);

preorder(p,n.left);

preorder(p,n.right);

}

return p;

}

//clear the tree

void clear()

{

root=null;

height = 0;

noOfNodes = 0;

}

//is empty

boolean isEmpty()

{ if(root == null) return true;

return false;

}

}

**Filename: FXML.fxml**

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.TextArea?>

<?import javafx.scene.control.TextField?>

<?import javafx.scene.effect.Glow?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.layout.BorderPane?>

<?import javafx.scene.layout.ColumnConstraints?>

<?import javafx.scene.layout.GridPane?>

<?import javafx.scene.layout.RowConstraints?>

<?import javafx.scene.layout.VBox?>

<BorderPane fx:id="rootPane" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity"

minWidth="-Infinity" prefHeight="500.0" prefWidth="900.0"

stylesheets="D:\\SNU\\Sem3\\CSD203\\lab\\assignment3\\src\\assignment3\\fxml.css"

xmlns="http://javafx.com/javafx/11.0.1" xmlns:fx="http://javafx.com/fxml/1"

fx:controller="assignment3.FXMLController">

<left>

<VBox spacing="40">

<VBox focusTraversable="true" spacing="5">

<TextField fx:id="value" prefColumnCount="6" prefHeight="25.0" promptText="Value" />

<Button mnemonicParsing="false" onAction="#insertButton" prefHeight="35.0" prefWidth="90.0" styleClass="control-button" text="Insert" />

<Button mnemonicParsing="false" onAction="#searchButton" prefHeight="35.0" prefWidth="90.0" styleClass="control-button" text="Search" />

<Button mnemonicParsing="false" onAction="#deleteButton" prefHeight="35.0" prefWidth="90.0" styleClass="control-button" text="Delete" />

<Button mnemonicParsing="false" onAction="#clearButton" prefHeight="35.0" prefWidth="90.0" styleClass="control-button" text="Clear" />

</VBox>

<VBox spacing="5">

<Button mnemonicParsing="false" onAction="#inorderButton" prefHeight="35.0" prefWidth="90.0" styleClass="control-button" text="Inorder" />

<Button mnemonicParsing="false" onAction="#preorderButton" prefHeight="35.0" prefWidth="90.0" styleClass="control-button" text="Preorder" />

<Button mnemonicParsing="false" onAction="#postorderButton" prefHeight="35.0" prefWidth="90.0" styleClass="control-button" text="Postorder" />

</VBox>

<VBox spacing="5">

<Button mnemonicParsing="false" onAction="#sortASCButton" prefHeight="35.0" prefWidth="90.0" styleClass="control-button" text="Sort ASC" />

<Button mnemonicParsing="false" onAction="#sortDESCButton" prefHeight="35.0" prefWidth="90.0" styleClass="control-button" text="Sort DESC" />

</VBox>

</VBox>

</left>

<right>

</right>

<top>

<GridPane alignment="CENTER\_LEFT">

<columnConstraints>

<ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" />

</columnConstraints>

<rowConstraints>

<RowConstraints maxHeight="-Infinity" minHeight="-Infinity" prefHeight="25.0" vgrow="NEVER" />

</rowConstraints>

<children>

<AnchorPane GridPane.hgrow="ALWAYS" GridPane.valignment="CENTER" GridPane.vgrow="NEVER">

<children>

<TextArea fx:id="valueList" editable="false" maxHeight="-Infinity" maxWidth="1.7976931348623157E308" minHeight="-Infinity" prefHeight="20.0" prefRowCount="1" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0" />

</children>

</AnchorPane>

</children>

</GridPane>

</top>

<bottom>

<GridPane alignment="CENTER\_LEFT">

<columnConstraints>

<ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" />

</columnConstraints>

<rowConstraints>

<RowConstraints maxHeight="-Infinity" minHeight="-Infinity" prefHeight="25.0" vgrow="NEVER" />

</rowConstraints>

<children>

<AnchorPane GridPane.hgrow="ALWAYS" GridPane.valignment="CENTER" GridPane.vgrow="NEVER">

<children>

<TextArea fx:id="counters" editable="false" maxHeight="-Infinity" maxWidth="1.7976931348623157E308" minHeight="-Infinity" prefHeight="20.0" prefRowCount="1" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0" />

</children>

</AnchorPane>

</children>

</GridPane>

</bottom>

</BorderPane>

***Output Test cases:***

1. Insert operation : 20, 10, 15, 30, 35, 25, 8, 12, 26, 22, 24
2. Delete operation: delete 20
3. Search Operation: find 12, find 20
4. Print operation: print sorted list (ascending), print sorted list (descending)
5. Traversal operation: Inorder traversal, Preorder traversal, Postorder traversal
6. Clear operation: clear the interface

**Note**:

1. The height and number of nodes counters, the status of delete and search operations and the result of search and traversal operations appear in the bottom bar.
2. Top bar shows the sequence of commands given by the user :-

Insert: i

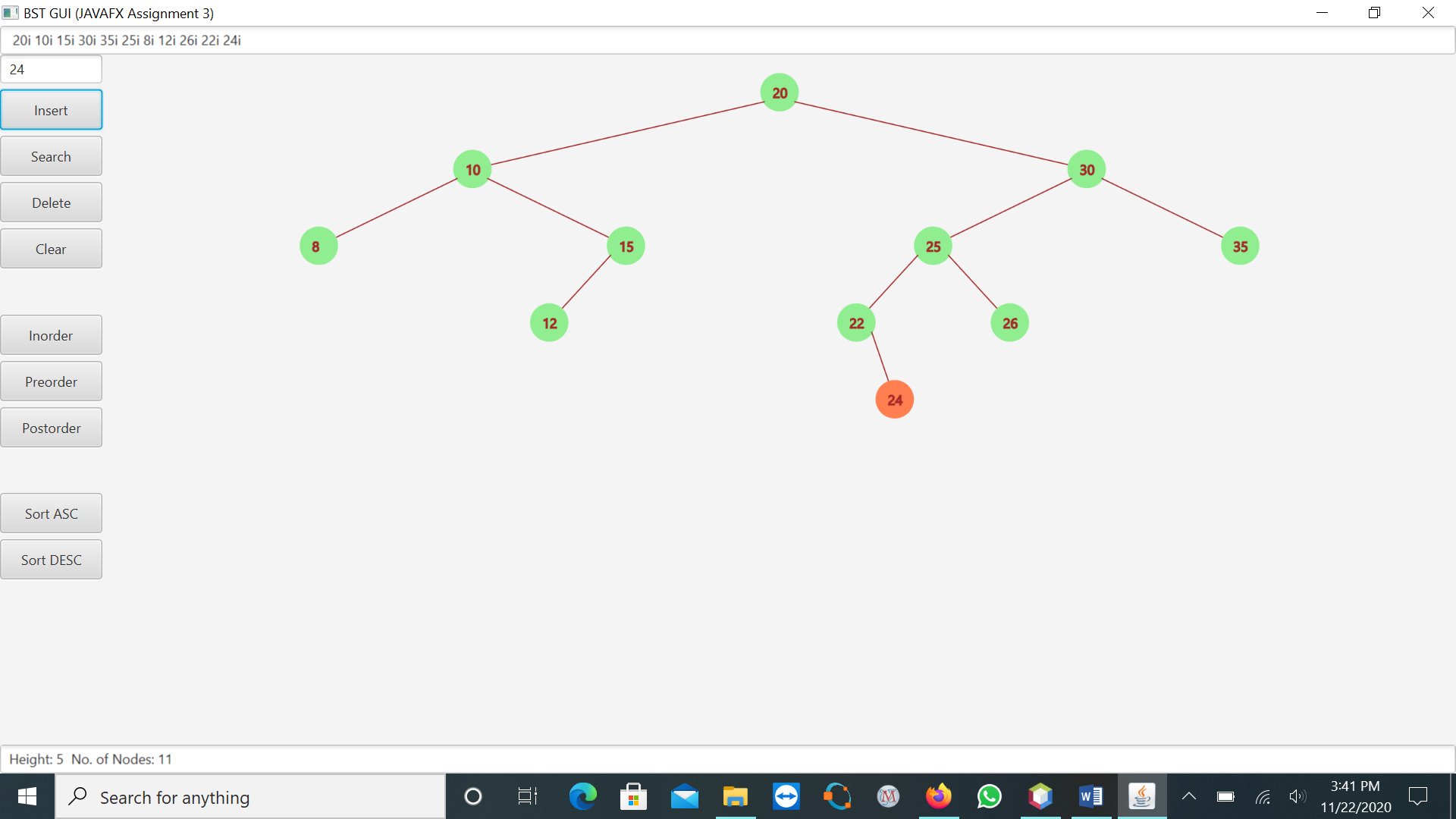
Delete: d

Search: s

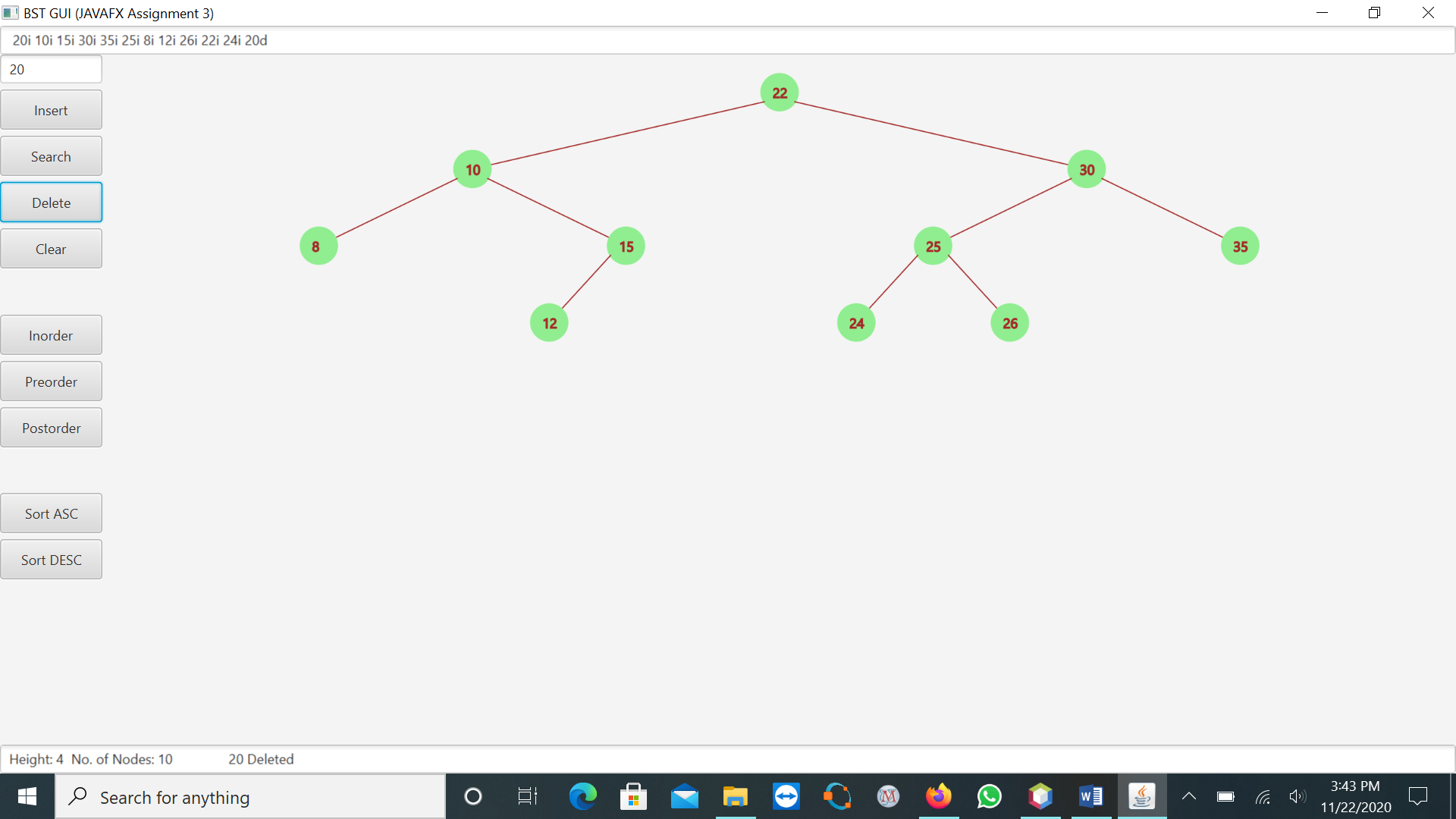
1. The node that is inserted/ searched (if found) id highlighted (CORAL/CYAN)
2. Resizing the window adjusts the tree accordingly.

**Output:**

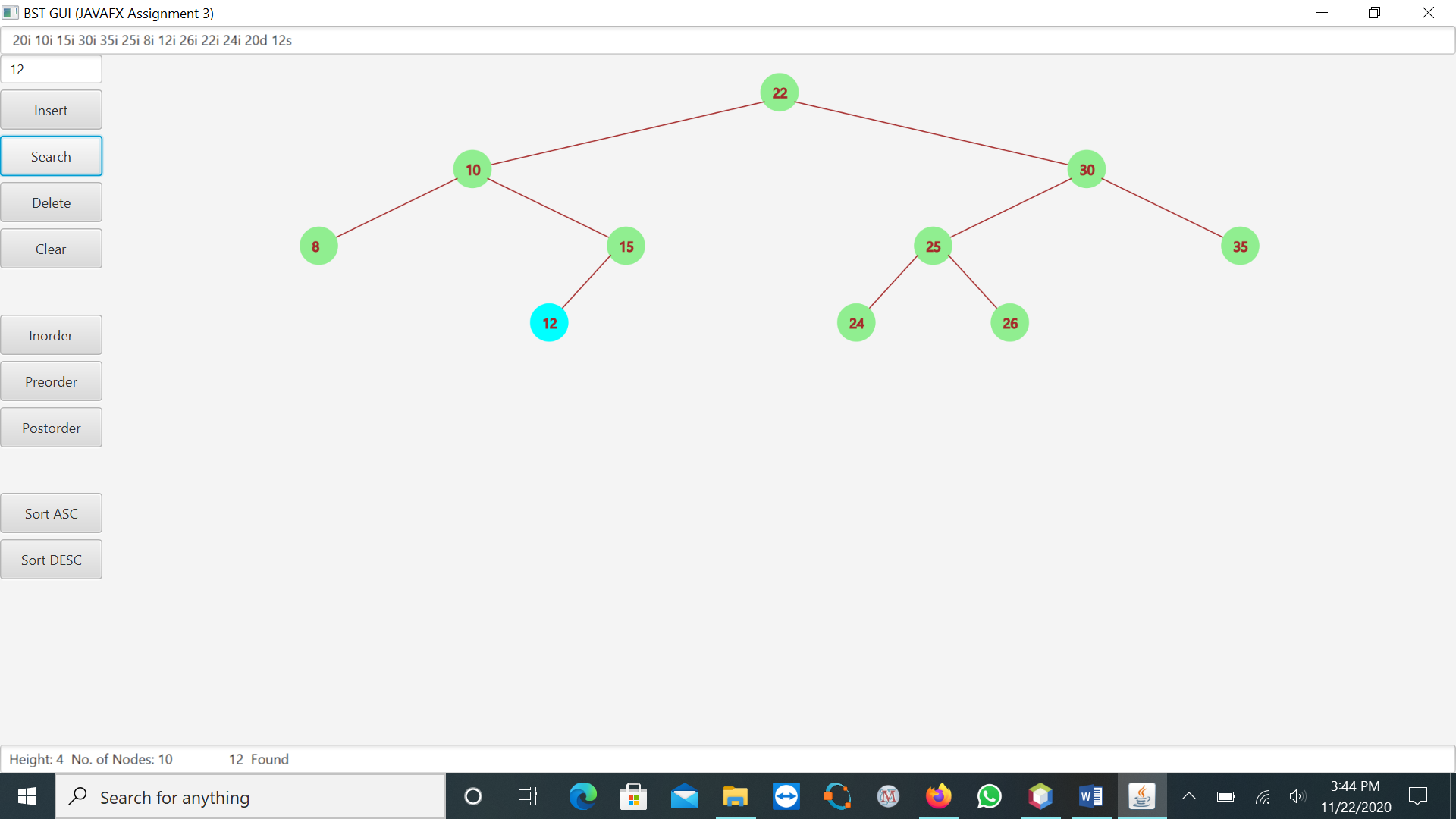
* 1. Insert operation

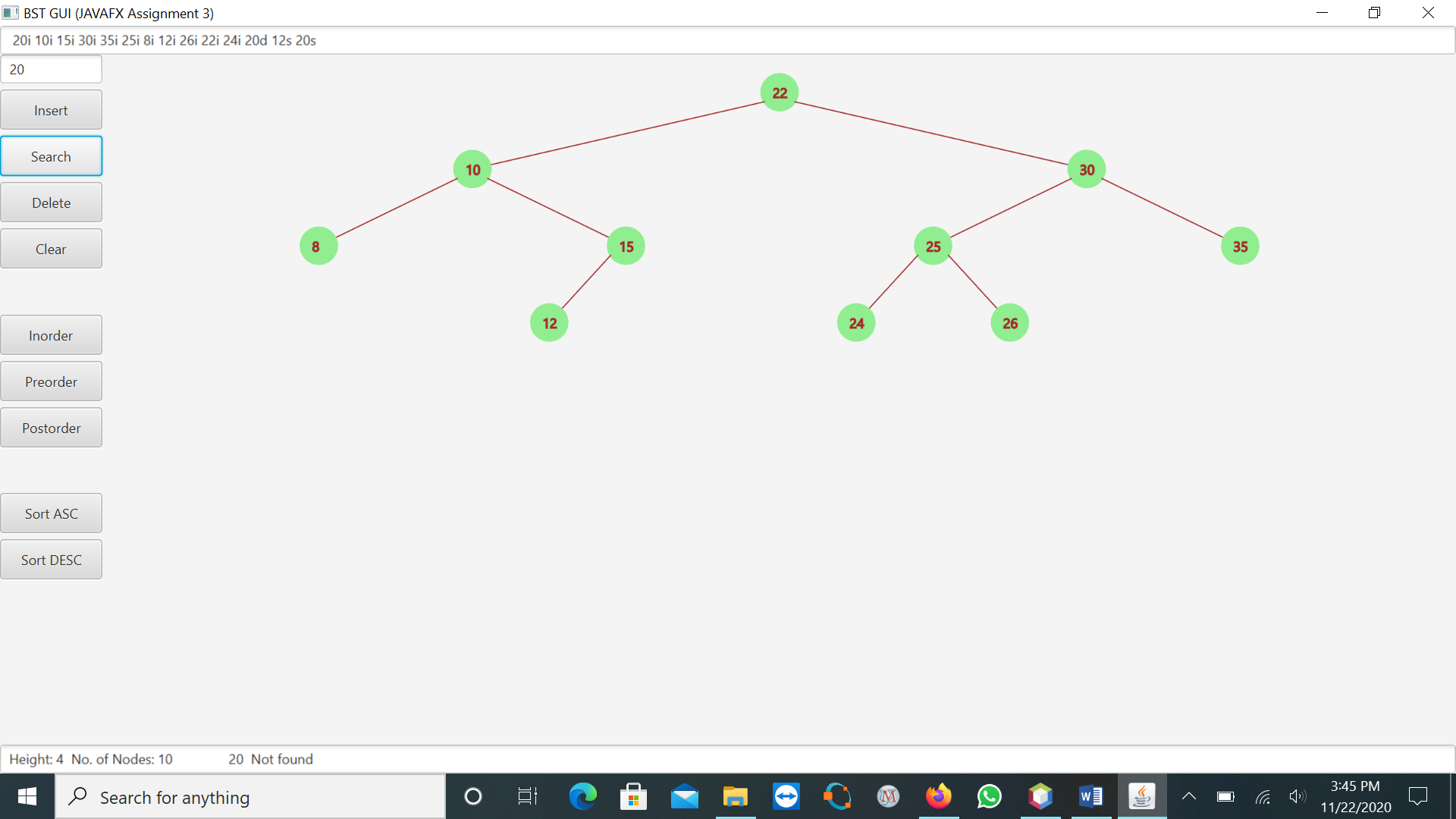


* 1. Delete operation

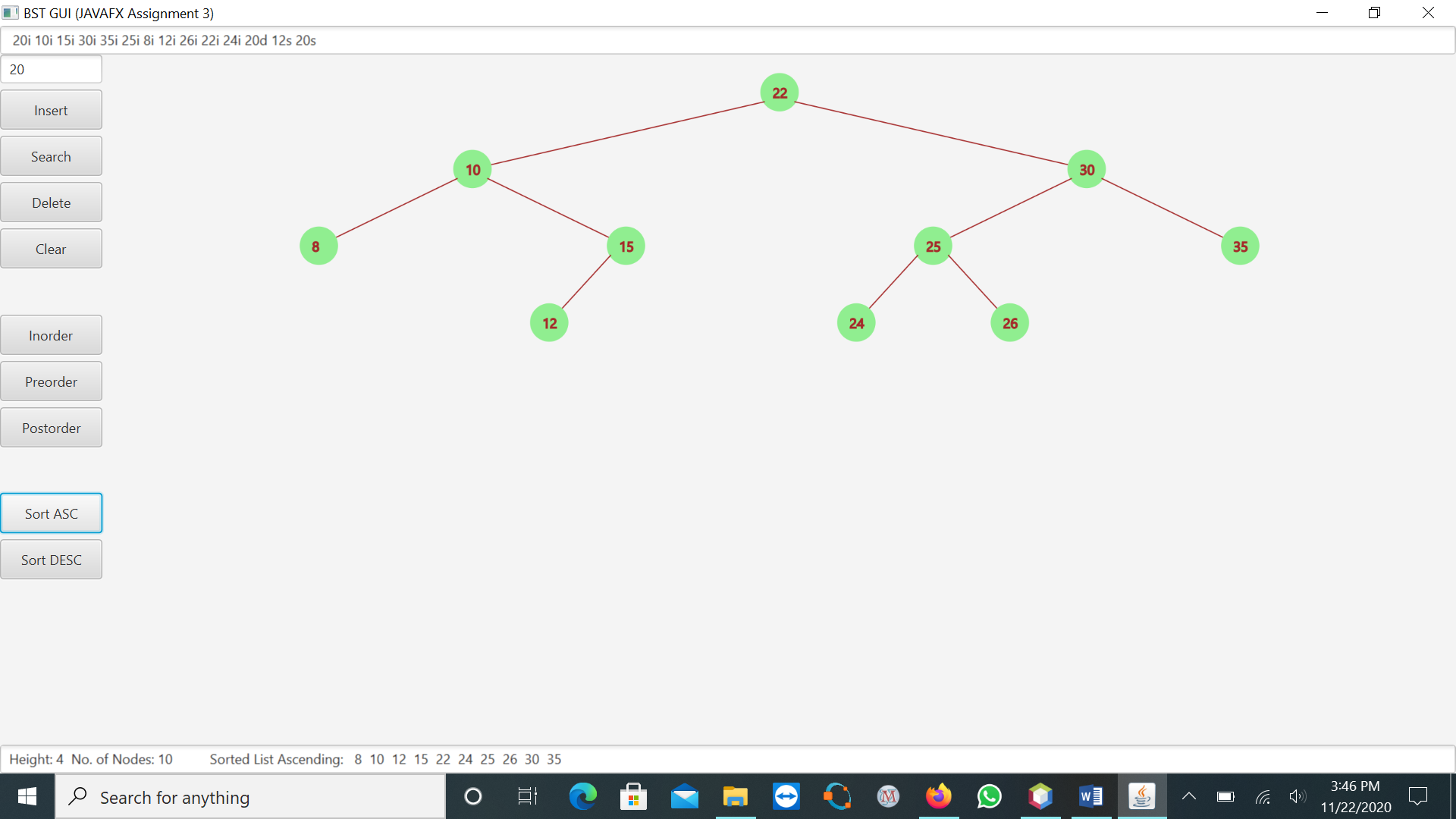


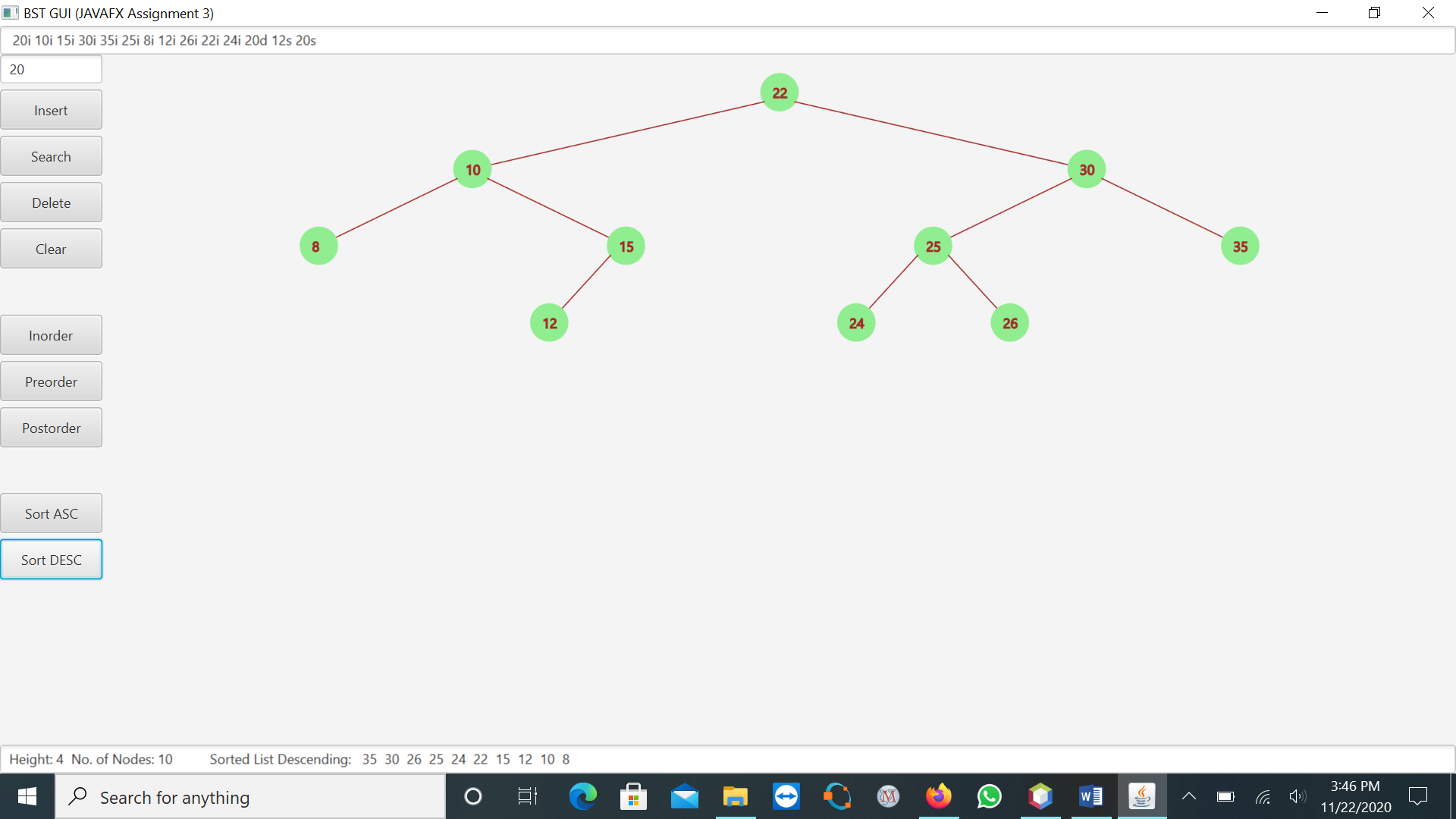
* 1. Search operation



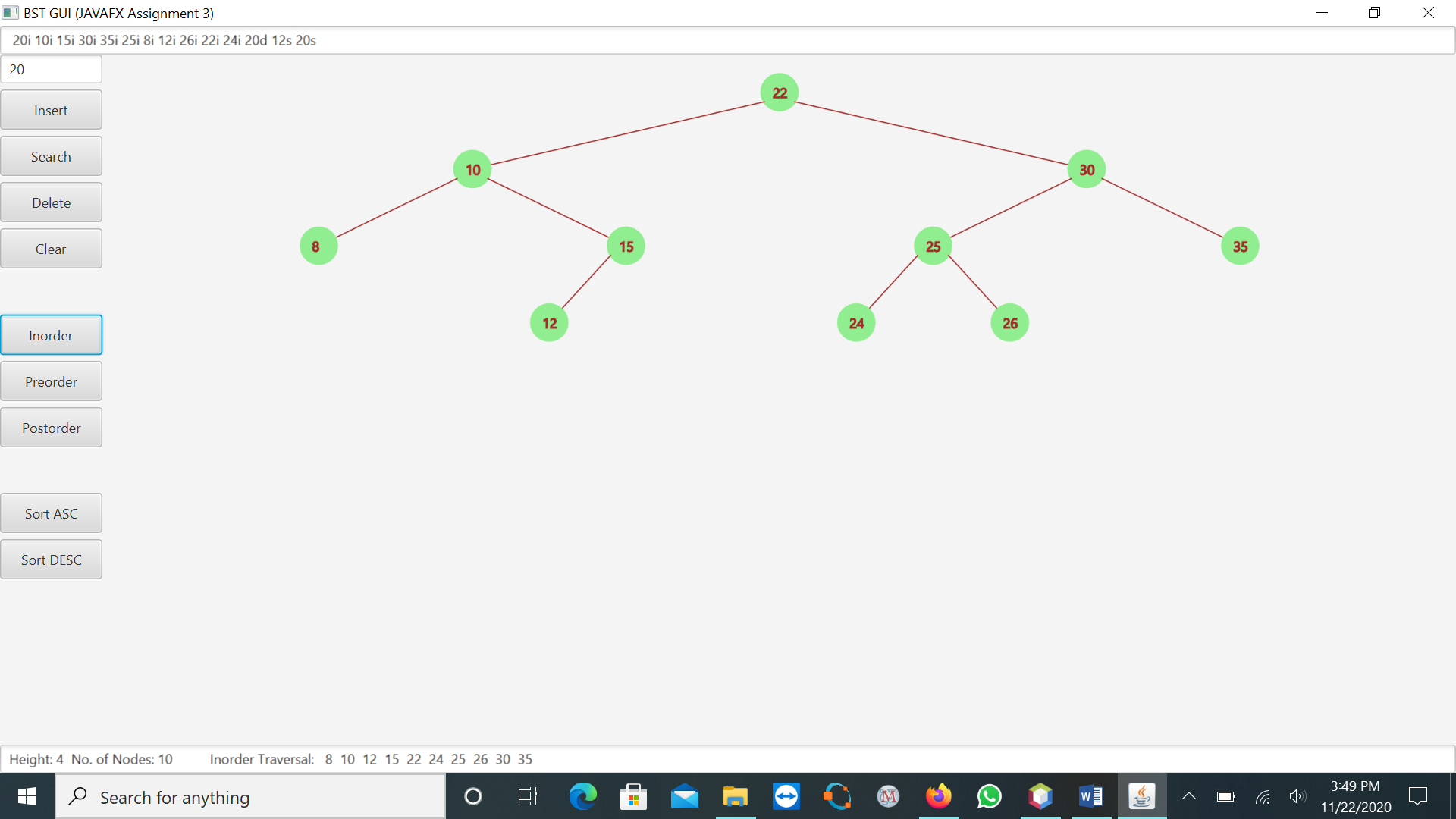


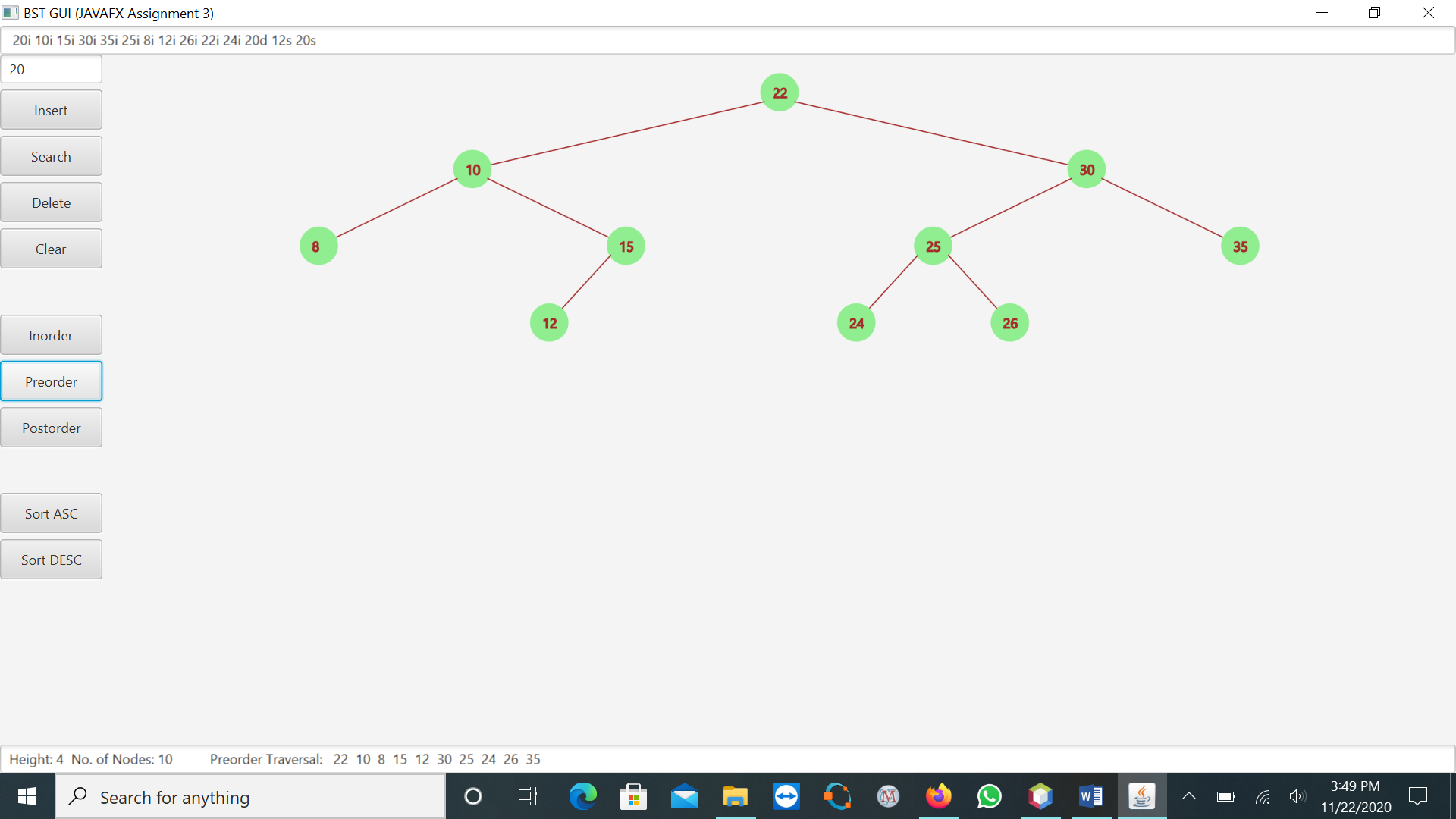
d. Print operation

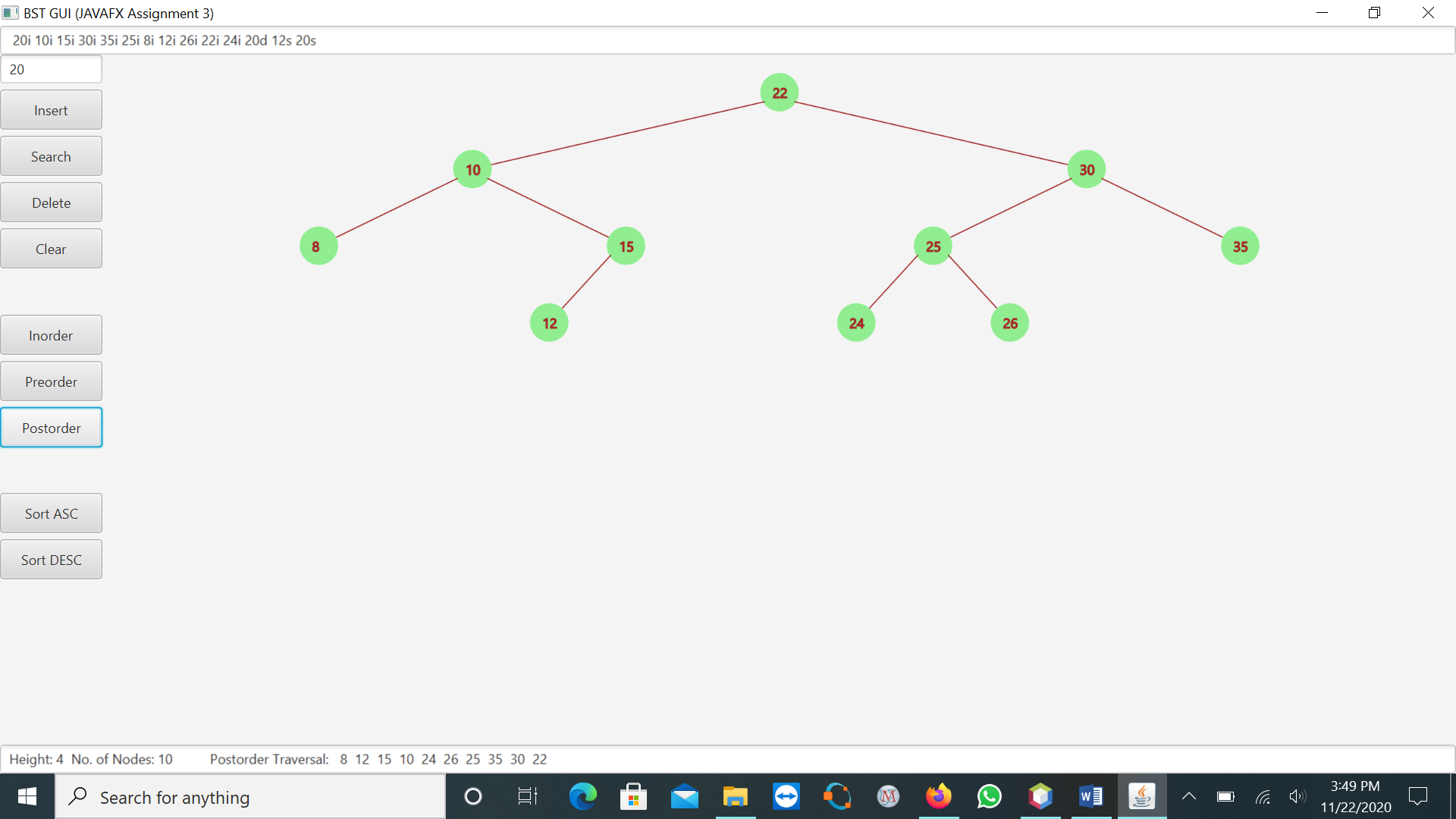




1. Traversal operation







1. Clear operation

