МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное
учреждение высшего
образования
НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий
Кафедра информатики и систем управления

ОТЧЕТ
по лабораторной работе
по дисциплине:
Алгоритмы и структуры данных

РУКОВОДИТЕЛЬ:
Капранов С.Н.
_____
(подпись)
(фамилия, и.,о.)

СТУДЕНТ:
Еричев Д.А.
_____
(подпись)
(фамилия, и.,о.)
_____
(шифр группы)

Работа защищена «___» _____
С оценкой _____

Нижний Новгород 2022

Задача 10.
Дано бинарное дерево. Определить какие поддеревья являются пирамидами.

1)Текст Программы:
Файл Main:

```
package Laba3;
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
public class Main {
    public static void main(String[] args) throws
FileNotFoundException {
        BST<Integer> tree = new BST<>();
        readElementsInFile(tree,
"/home/denis/JavaPrograms/src/Laba3/file.txt");
        System.out.println("Бинарное дерево:");
        tree.print();
        System.out.println();
        tree.find();
    }

    static void readElementsInFile(BST<Integer> tree, String path)
throws FileNotFoundException {
        Scanner scanner = new Scanner(new File(path));
        while(scanner.hasNextInt()) {
            tree.add(scanner.nextInt());
        }
    }
}
```

Файл BST:

```
package Laba3;
public class BST<T> {
    BST() {root = null;}
    private class Node<T>{ //класс элемента дерева
        Node() {};
        Node(int value) {
            this.value = value;
            right = null;
            left = null;
        }
        public int value; //значение элемента
        Node<T> left = null;    //левая ветвь
        Node<T> right = null;   //правая ветвь
        public int getValue() {
            return value;
        }
        public void setValue(int value) {
            this.value = value;
        }
    }
    private Node<T> root = null;    //корень дерева
```

```
    private Node addNode(Node<T> node, int value) {
        if(node == null) {
          node = new Node<>(value);
          node.left = node.right = null;
        }
        else if(node.value < value){
            node.right = addNode(node.right, value);
        }
        else if(node.value > value){
            node.left = addNode(node.left, value);
        }
        return node;
    }
    public void add(int value) {
        root = addNode(root, value);
    }   //первый вызов для добавления элемента
    private void printTree(Node<T> node) {        //Обход дерева
        if(node == null) return;    //если дерево пустое ничего не
делаем
        printTree(node.left);       //заходим в левую(меньшую)
ветку
        printTree(node.right);      //заходим в правую(большую)
ветку
        System.out.print(node.getValue() + " ");    //выводим
значения(по возрастанию)
    }
    public void print() {
        printTree(root);
    }   //первый вызов для вывода элементов дерева
    private void delTree(Node<T> root) {    //Удаление всех узлов
(очистка дерева)
        if(root != null) {          //если узел не пустой
            delTree(root.left);     //удаляем левую ветку
            delTree(root.right);    //удаляем правую ветку
        }
    }
    public void clear() {   //первый вызов для очистки дерева
        if(root != null) {
            delTree(root);
            root = null;
        }
        System.out.println("Элементы удалены.");
    }
    public int counterElements(Node<T> node) {
        if(node == null) {
            return 0;
        }
        return counterElements(node.right) +
counterElements(node.left) + 1;
    }
    public int findPyramid(Node<T> node, int countElements) {
        if((node.left == null && node.right == null) || node.left
== null || node.right == null){
```

```
            return 1;
        }
        else if(countElements % 2 == 0 ) {
            findPyramid(node.right,counterElements(node.right));
            findPyramid(node.left,counterElements(node.left));
            return 1;
        }
        else if(node.left != null && node.right != null){
            findPyramid(node.left, countElements);
            findPyramid(node.right,countElements);
            if(node != root) {
                System.out.print(node.getValue() +" ");
            }
            if(node.right.right == null && node.left.left == null)
{
                System.out.print(node.left.getValue()+" ");
                System.out.print(node.right.getValue()+" ");
            }
            System.out.println();
        }
        return 1;
    }
    private void findThreePyramid(Node<T> node) {        //метод
для поиска пирамид поддеревьев
        if(node != null && node.left != null && node.right !=
null) {  //если у узла есть правая и левая ветвь
            findThreePyramid(node.left);     //идем в левую ветвь
            findThreePyramid(node.right);    //идем в правую ветвь
            if(node != root) {
                System.out.println(node.getValue() + " " +
node.left.getValue() + " " + node.right.getValue()); //выводим эту
пирамиду
            }
        }
    }
    public void find() {   //первый вызов ф-ии для нахождения
пирамид
        System.out.println("Поддеревья бинарного дерева, которые
являются пирамидами:");

        System.out.println("Пирамиды состоящие из 3-х
элементов:");
        findThreePyramid(root);
        System.out.println();

        System.out.println("Пирамиды левого поддерева:");
        findPyramid(root.left,counterElements(root.left));
        System.out.println();

        System.out.println("Пирамиды правого поддерева:");
        findPyramid(root.right,counterElements(root.right));
```

```
        }
}
```

2)Описание функций:
static void readElementsInFile(BST<Integer> tree, String path): считывание элементов дерева из файла

private Node addNode(Node<T> node, int value) : добавление элементов в бинарное дерево
private void printTree(Node<T> node) : обход дерева(вывод в консоль)
public void clear(): очистка дерева
public int counterElements(Node<T> node): подсчет элементов в поддереве
public int findPyramid(Node<T> node, int countElements): поиск полных пирамид
private void findThreePyramid(Node<T> node) : поиск подПирамид(состоящих из трех элементов)
public void find(): вызов функции findThreePyramid и вывод с обозначениями

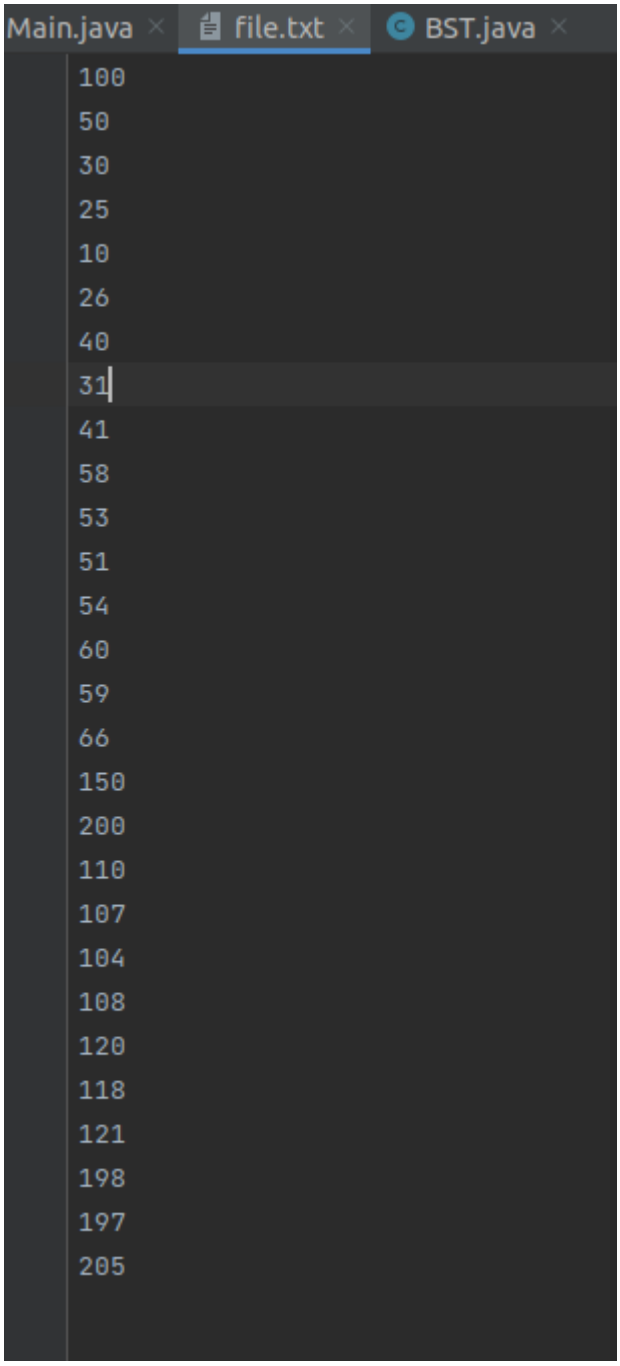
Вывод в консоль:

```
Бинарное дерево:
10 26 25 31 41 40 30 51 54 53 59 66 60 58 50 104 108 107 118 121 120 110 197 198 205 200 150 100
Поддеревья бинарного дерева, которые являются пирамидами:
Пирамиды состоящие из 3-х элементов:
25 10 26
40 31 41
30 25 40
53 51 54
60 59 66
58 53 60
50 30 58
107 104 108
120 118 121
110 107 120
200 198 205
150 110 200

Пирамиды левого поддерева:
25 10 26
40 31 41
30
53 51 54
60 59 66
58
50

Пирамиды правого поддерева:
107 104 108
120 118 121
110
```

Считывание из файла:


Main.java
file.txt
BST.java
100
50
30
25
10
26
40
31
41
58
53
51
54
60
59
66
150
200
110
107
104
108
120
118
121
198
197
205