VIJAY M

22IT125 Day-3

CODING PRACTICES AND PROBLEM

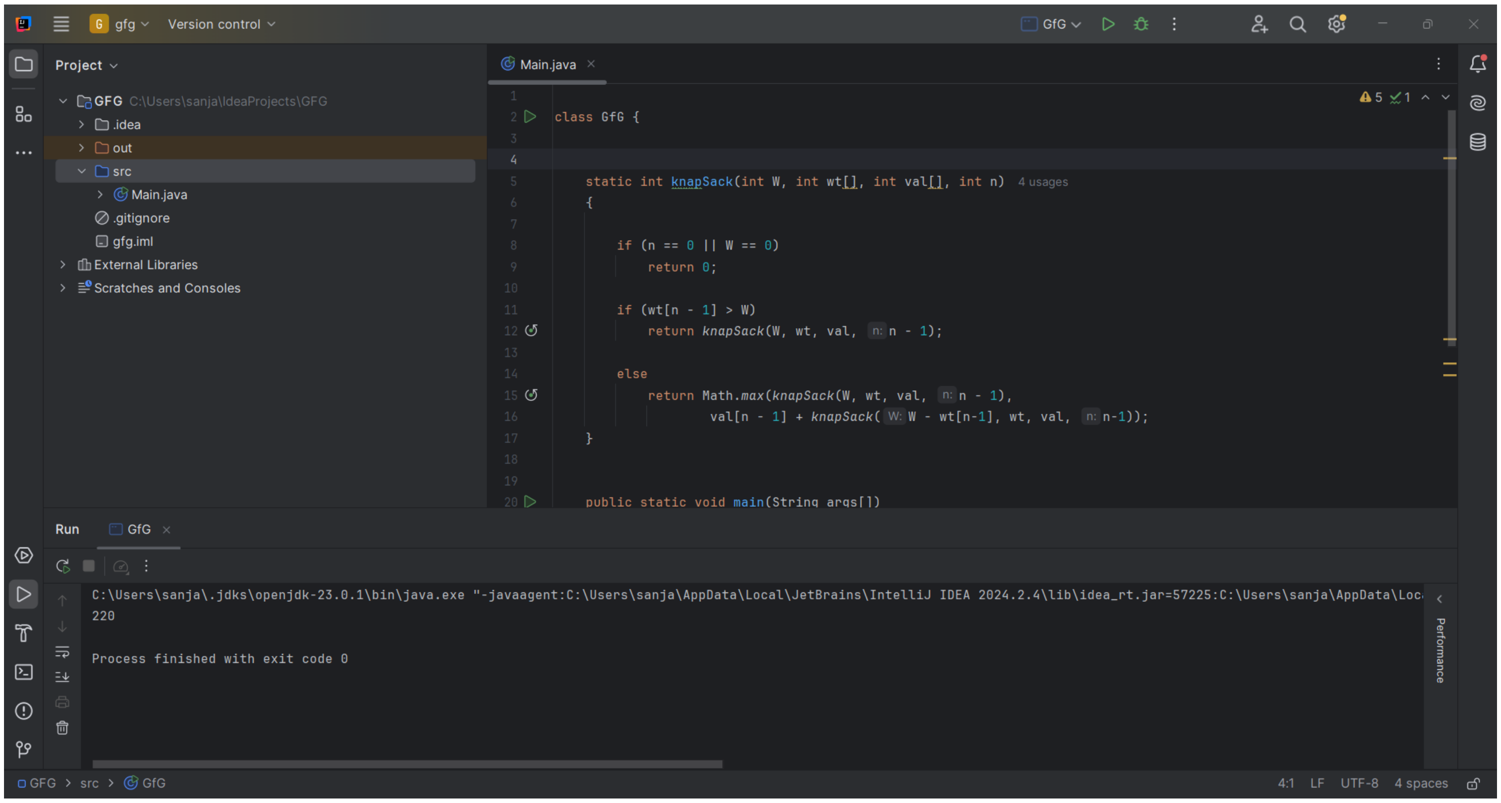
1.0-1 Knapsack Problem

Time Complexity : O(2N)

Solution:

```
class GfG {

   static int knapSack(int W, int wt[], int val[], int n)
   {

       if (n == 0 || W == 0)
           return 0;

       if (wt[n - 1] > W)
           return knapSack(W, wt, val, n - 1);

       else
           return Math.max(knapSack(W, wt, val, n - 1),
                   val[n - 1] + knapSack(W - wt[n-1], wt, val, n-1));
   }

   public static void main(String args[])
   {
       int profit[] = new int[] { 60, 100, 120 };
       int weight[] = new int[] { 10, 20, 30 };
       int W = 50;
       int n = profit.length;
       System.out.println(knapSack(W, weight, profit, n));
   }
}
```

Output:



```
class GfG {

    static int knapSack(int W, int wt[], int val[], int n)  4 usages
    {

        if (n == 0 || W == 0)
            return 0;

        if (wt[n - 1] > W)
            return knapSack(W, wt, val, n: n - 1);

        else
            return Math.max(knapSack(W, wt, val, n: n - 1),
                    val[n - 1] + knapSack( W: W - wt[n-1], wt, val, n: n-1));
    }


    public static void main(String args[])
```

```
C:\Users\sanja\.jdks\openjdk-23.0.1\bin\java.exe "-javaagent:C:\Users\sanja\AppData\Local\JetBrains\IntelliJ IDEA 2024.2.4\lib\idea_rt.jar=57225:C:\Users\sanja\AppData\Loc
220

Process finished with exit code 0
```

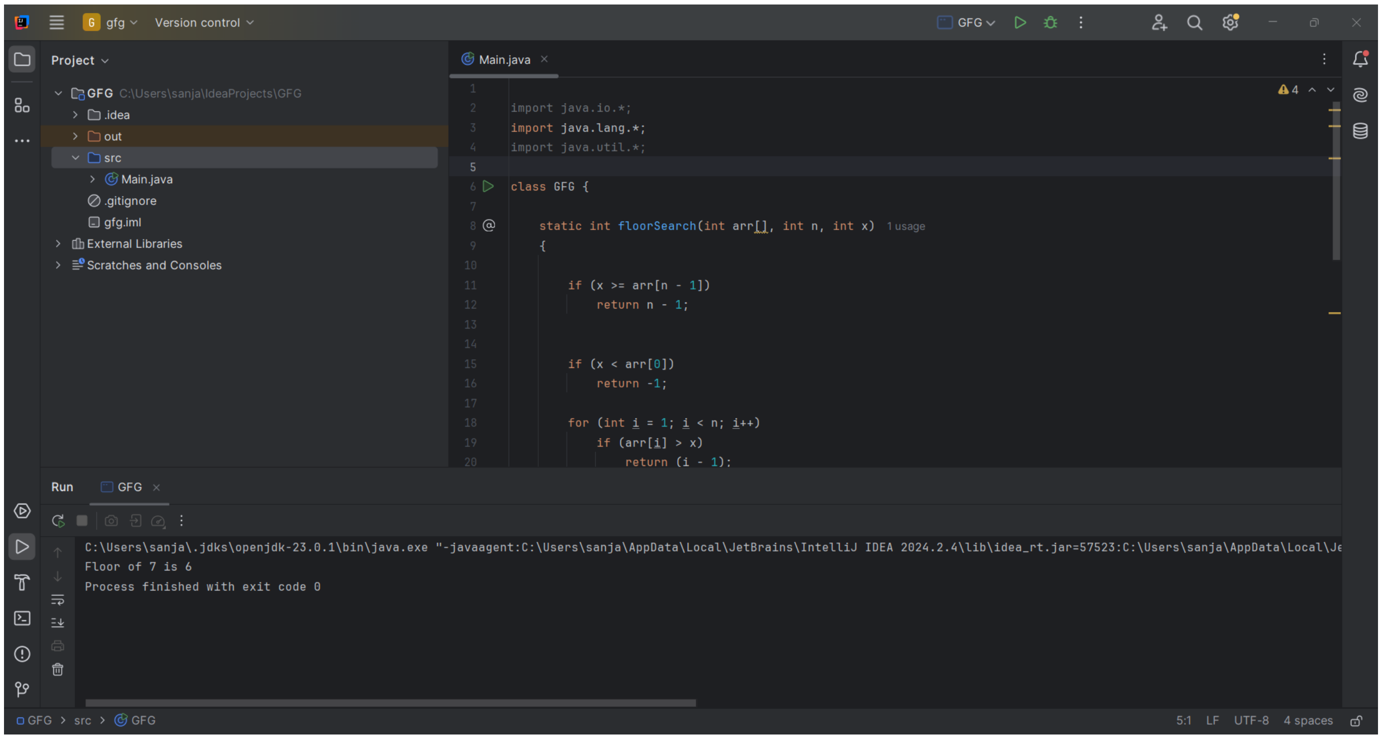
2.Floor in Sorted Array:

Time Complexity : O(N)

Solution:

```
import java.io.*;
import java.lang.*;
import java.util.*;

class GFG {

    static int floorSearch(int arr[], int n, int x)
    {

        if (x >= arr[n - 1])
            return n - 1;
        if (x < arr[0])
            return -1;

        for (int i = 1; i < n; i++)
            if (arr[i] > x)
                return (i - 1);

        return -1;
    }

    public static void main(String[] args)
    {
        int arr[] = { 1, 2, 4, 6, 10, 12, 14 };
        int n = arr.length;
        int x = 7;
        int index = floorSearch(arr, n - 1, x);
        if (index == -1)
            System.out.print("Floor of " + x
                    + " doesn't exist in array ");
        else
            System.out.print("Floor of " + x + " is "
                    + arr[index]);
    }
}
```

Output:



```java
import java.io.*;
import java.lang.*;
import java.util.*;

class GFG {

    static int floorSearch(int arr[], int n, int x)
    {

        if (x >= arr[n - 1])
            return n - 1;


        if (x < arr[0])
            return -1;

        for (int i = 1; i < n; i++)
            if (arr[i] > x)
                return (i - 1);
```

```
C:\Users\sanja\.jdks\openjdk-23.0.1\bin\java.exe "-javaagent:C:\Users\sanja\AppData\Local\JetBrains\IntelliJ IDEA 2024.2.4\lib\idea_rt.jar=57523:C:\Users\sanja\AppData\Local\Je
Floor of 7 is 6
Process finished with exit code 0
```

3.Check Equal Arrays:

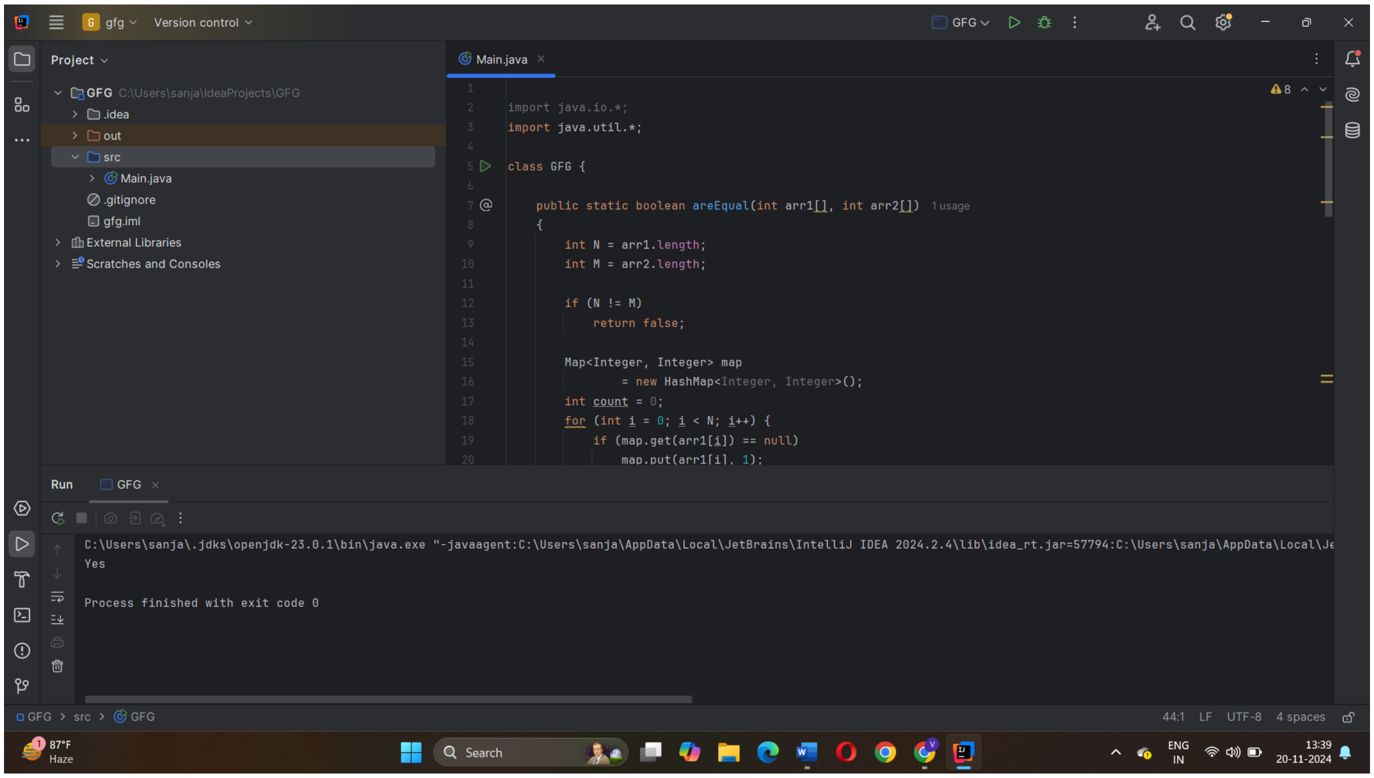
Time Complexity : O(N*log(N))

Solution:

```java
import java.io.*;
import java.util.*;

class GFG {

    public static boolean areEqual(int arr1[], int arr2[])
    {
        int N = arr1.length;
        int M = arr2.length;

        if (N != M)
            return false;
    Map<Integer, Integer> map
            = new HashMap<Integer, Integer>();
    int count = 0;
    for (int i = 0; i < N; i++) {
        if (map.get(arr1[i]) == null)
            map.put(arr1[i], 1);
        else {
            count = map.get(arr1[i]);
            count++;
            map.put(arr1[i], count);
        }
    }

    for (int i = 0; i < N; i++) {



        if (!map.containsKey(arr2[i]))
            return false;

        if (map.get(arr2[i]) == 0)
            return false;

        count = map.get(arr2[i]);
        --count;
        map.put(arr2[i], count);
    }

    return true;
}

public static void main(String[] args)
{
    int arr1[] = { 3, 5, 2, 5, 2 };
    int arr2[] = { 2, 3, 5, 5, 2 };
```

```
        if (areEqual(arr1, arr2))
            System.out.println("Yes");
        else
            System.out.println("No");
    }
}
```

Output:



```
import java.io.*;
import java.util.*;

class GFG {

    public static boolean areEqual(int arr1[], int arr2[])
    {
        int N = arr1.length;
        int M = arr2.length;

        if (N != M)
            return false;

        Map<Integer, Integer> map
                = new HashMap<Integer, Integer>();
        int count = 0;
        for (int i = 0; i < N; i++) {
            if (map.get(arr1[i]) == null)
                map.put(arr1[i], 1);
```

```
C:\Users\sanja\.jdks\openjdk-23.0.1\bin\java.exe "-javaagent:C:\Users\sanja\AppData\Local\JetBrains\IntelliJ IDEA 2024.2.4\lib\idea_rt.jar=57794:C:\Users\sanja\AppData\Local\Je
Yes

Process finished with exit code 0
```

4.Palindrome Linked List:

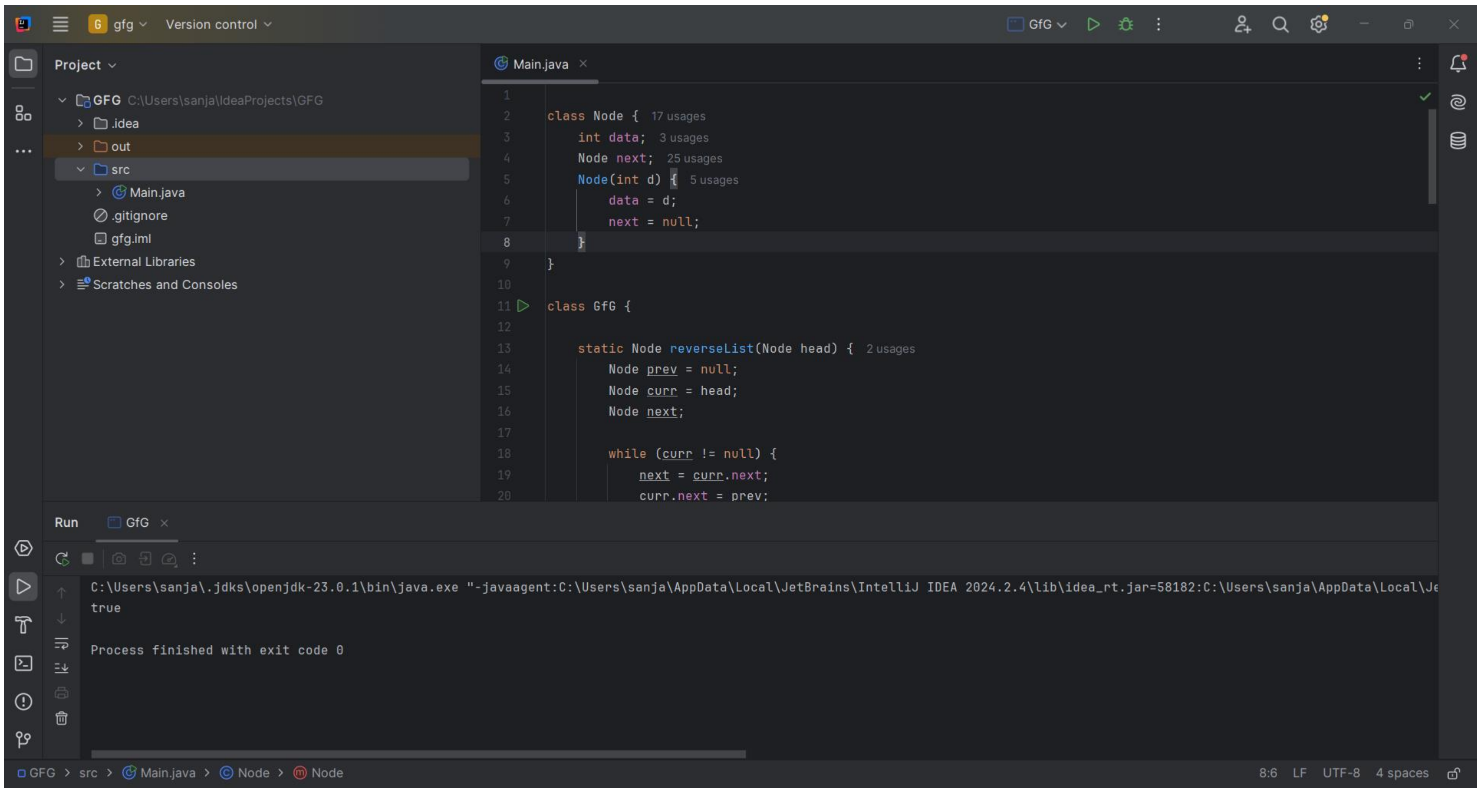
Time Complexity : O(n)

Solution:

```
class Node {
    int data;
    Node next;
    Node(int d) {
        data = d;
        next = null;
    }
}

class GfG {

    static Node reverseList(Node head) {
        Node prev = null;
        Node curr = head;
        Node next;

        while (curr != null) {
            next = curr.next;
            curr.next = prev;
            prev = curr;
            curr = next;
        }
        return prev;
    }

    static boolean isIdentical(Node n1, Node n2) {
        while (n1 != null && n2 != null) {
            if (n1.data != n2.data)
                return false;
            n1 = n1.next;
        n2 = n2.next;
    }
    return true;
}

static boolean isPalindrome(Node head) {
    if (head == null || head.next == null)
        return true;

    Node slow = head, fast = head;

    while (fast.next != null
            && fast.next.next != null) {
        slow = slow.next;
        fast = fast.next.next;
    }

    Node head2 = reverseList(slow.next);
    slow.next = null;

    boolean ret = isIdentical(head, head2);

    head2 = reverseList(head2);
    slow.next = head2;

    return ret;
}

public static void main(String[] args) {

    Node head = new Node(1);
    head.next = new Node(2);
    head.next.next = new Node(3);
    head.next.next.next = new Node(2);
    head.next.next.next.next = new Node(1);
```

```
        boolean result = isPalindrome(head);

        if (result)
            System.out.println("true");
        else
            System.out.println("false");
    }
}
```

Output:



```
class Node {  17 usages
    int data;  3 usages
    Node next;  25 usages
    Node(int d) {  5 usages
        data = d;
        next = null;
    }
}

class GfG {

    static Node reverseList(Node head) {  2 usages
        Node prev = null;
        Node curr = head;
        Node next;

        while (curr != null) {
            next = curr.next;
            curr.next = prev;
```

```
C:\Users\sanja\.jdks\openjdk-23.0.1\bin\java.exe "-javaagent:C:\Users\sanja\AppData\Local\JetBrains\IntelliJ IDEA 2024.2.4\lib\idea_rt.jar=58182:C:\Users\sanja\AppData\Local\Je
true

Process finished with exit code 0
```

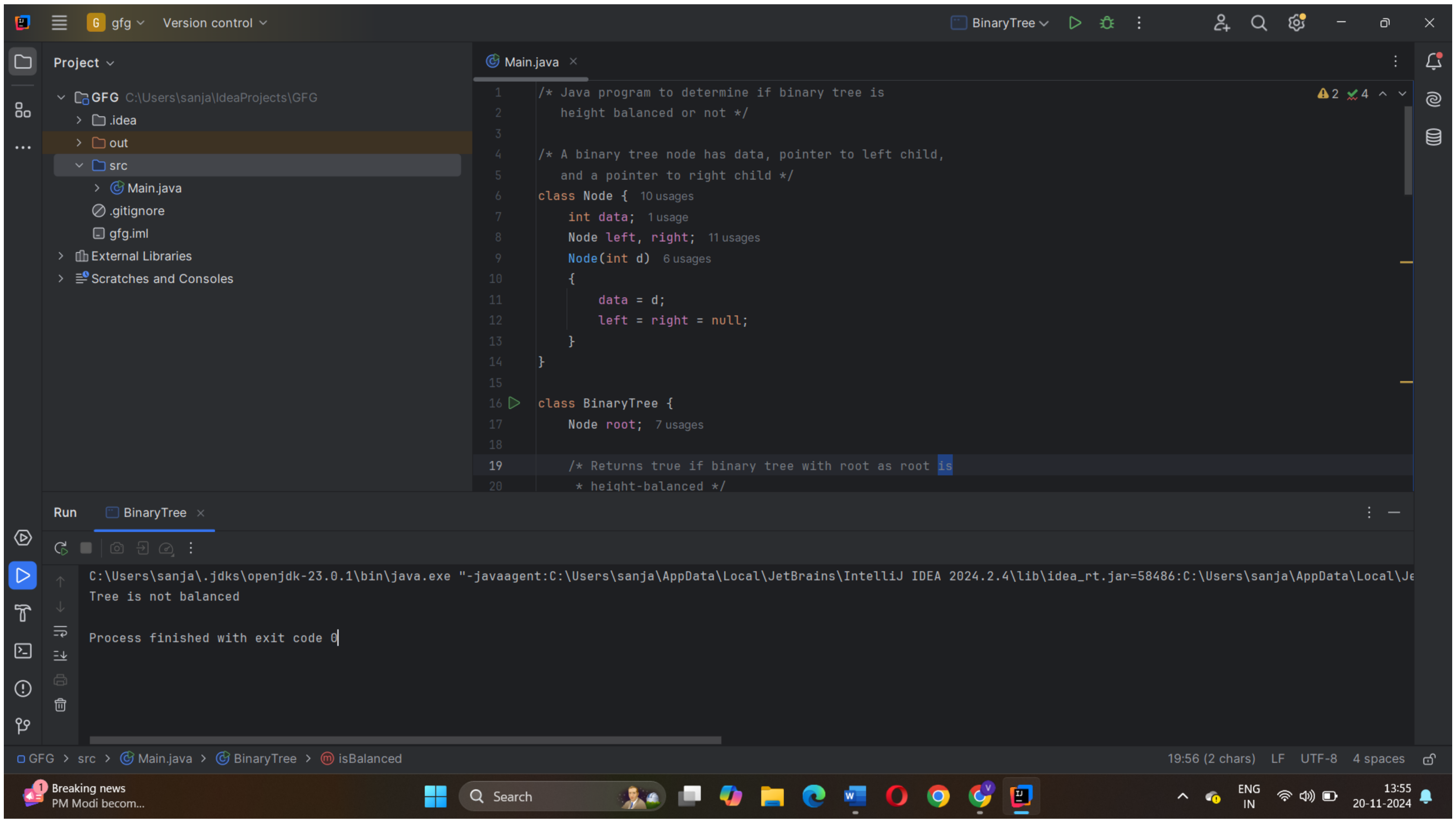
5.Balanced Tree Check:

Time Comlexity : O(n^2)

Solution:

```
class Node {
    int data;
    Node left, right;
    Node(int d)
    {
        data = d;
        left = right = null;
    }
}

class BinaryTree {
    Node root;


    boolean isBalanced(Node node)
    {

        if (node == null)
            return true;

        lh = height(node.left);
        rh = height(node.right);

        if (Math.abs(lh - rh) <= 1 && isBalanced(node.left)
                && isBalanced(node.right))
            return true;



        return false;
    }
    int height(Node node)
    {

        if (node == null)
            return 0;

        return 1
                + Math.max(height(node.left),
                height(node.right));
    }

    public static void main(String args[])
    {
        BinaryTree tree = new BinaryTree();
        tree.root = new Node(1);
        tree.root.left = new Node(2);
        tree.root.right = new Node(3);
        tree.root.left.left = new Node(4);
        tree.root.left.right = new Node(5);
        tree.root.left.left.left = new Node(8);

        if (tree.isBalanced(tree.root))
            System.out.println("Tree is balanced");
        else
            System.out.println("Tree is not balanced");
    }
}
```

Output:



```
/* Java program to determine if binary tree is
   height balanced or not */

/* A binary tree node has data, pointer to left child,
   and a pointer to right child */
class Node {
    int data;
    Node left, right;
    Node(int d)
    {
        data = d;
        left = right = null;
    }
}

class BinaryTree {
    Node root;

    /* Returns true if binary tree with root as root is
     * height-balanced */
```

```
C:\Users\sanja\.jdks\openjdk-23.0.1\bin\java.exe "-javaagent:C:\Users\sanja\AppData\Local\JetBrains\IntelliJ IDEA 2024.2.4\lib\idea_rt.jar=58486:C:\Users\sanja\AppData\Local\Je
Tree is not balanced

Process finished with exit code 0
```

6.Triplet Sum In Array:

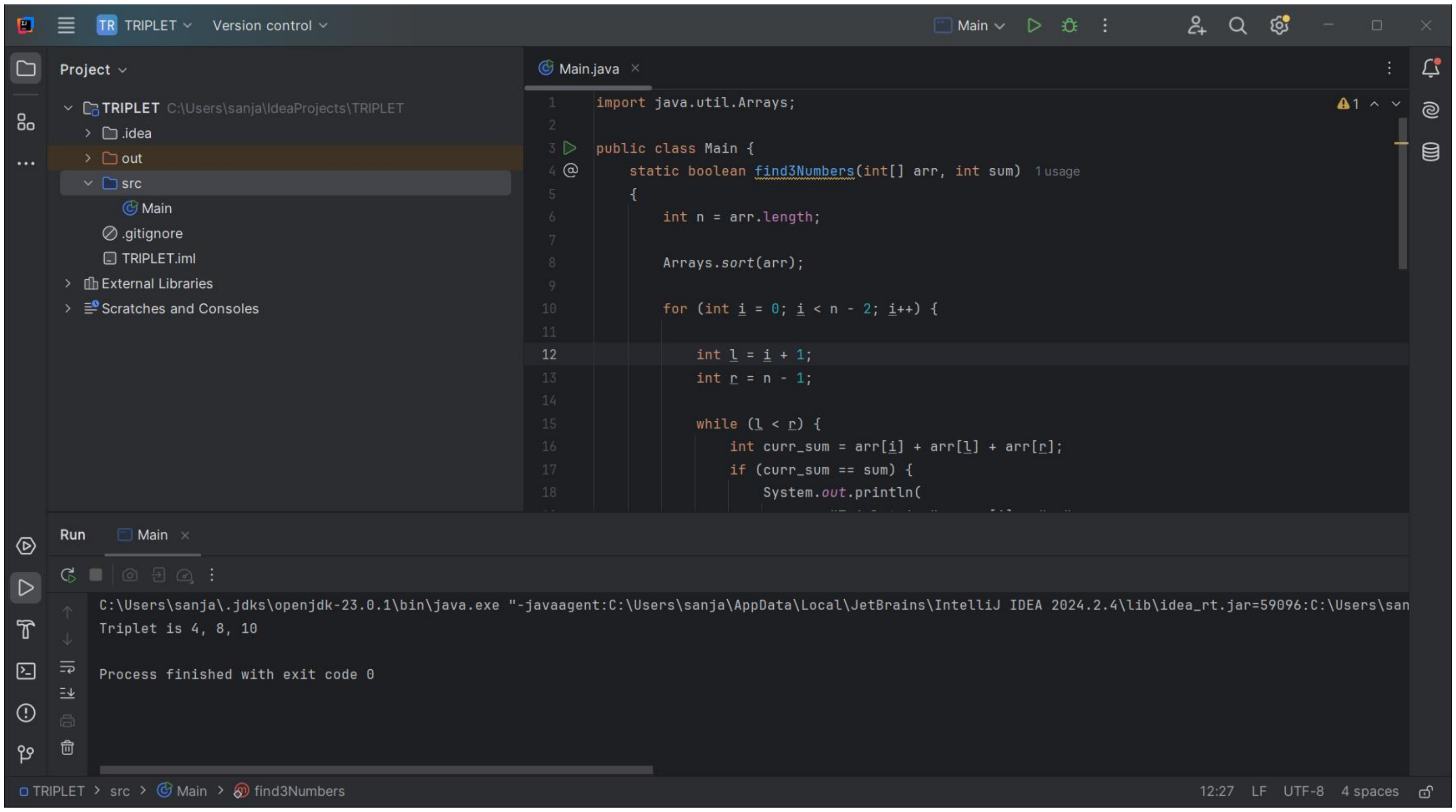
Time Complexity : O(n^2)

Solution:

```
import java.util.Arrays;
public class Main {
   static boolean find3Numbers(int[] arr, int sum)
   {
      int n = arr.length;
      Arrays.sort(arr);
      for (int i = 0; i < n - 2; i++) {
         int l = i + 1;
         int r = n - 1;
```

```
            while (l < r) {
                int curr_sum = arr[i] + arr[l] + arr[r];
                if (curr_sum == sum) {
                    System.out.println(
                        "Triplet is " + arr[i] + ", "
                        + arr[l] + ", " + arr[r]);
                    return true;
                }
                else if (curr_sum < sum) {
                    l++;
                }
                else {
                    r--;
                }
            }
        }
        return false;
    }
    public static void main(String[] args)
    {
        int[] arr = { 1, 4, 45, 6, 10, 8 };
        int sum = 22;
        find3Numbers(arr, sum);
    }
}
```

Output:



```java
import java.util.Arrays;

public class Main {
    static boolean find3Numbers(int[] arr, int sum)
    {
        int n = arr.length;

        Arrays.sort(arr);

        for (int i = 0; i < n - 2; i++) {

            int l = i + 1;
            int r = n - 1;

            while (l < r) {
                int curr_sum = arr[i] + arr[l] + arr[r];
                if (curr_sum == sum) {
                    System.out.println(
```

```
C:\Users\sanja\.jdks\openjdk-23.0.1\bin\java.exe "-javaagent:C:\Users\sanja\AppData\Local\JetBrains\IntelliJ IDEA 2024.2.4\lib\idea_rt.jar=59096:C:\Users\san
Triplet is 4, 8, 10

Process finished with exit code 0
```