NAME:-SHREYASI REJA

Question-1

Find a pair with the given sum in an array

Given an unsorted integer array, find a pair with the given sum in it.

For example

Input: nums = [8, 7, 2, 5, 3, 1]target = 10 Output: Pair found (8, 2)orPair found (7, 3)

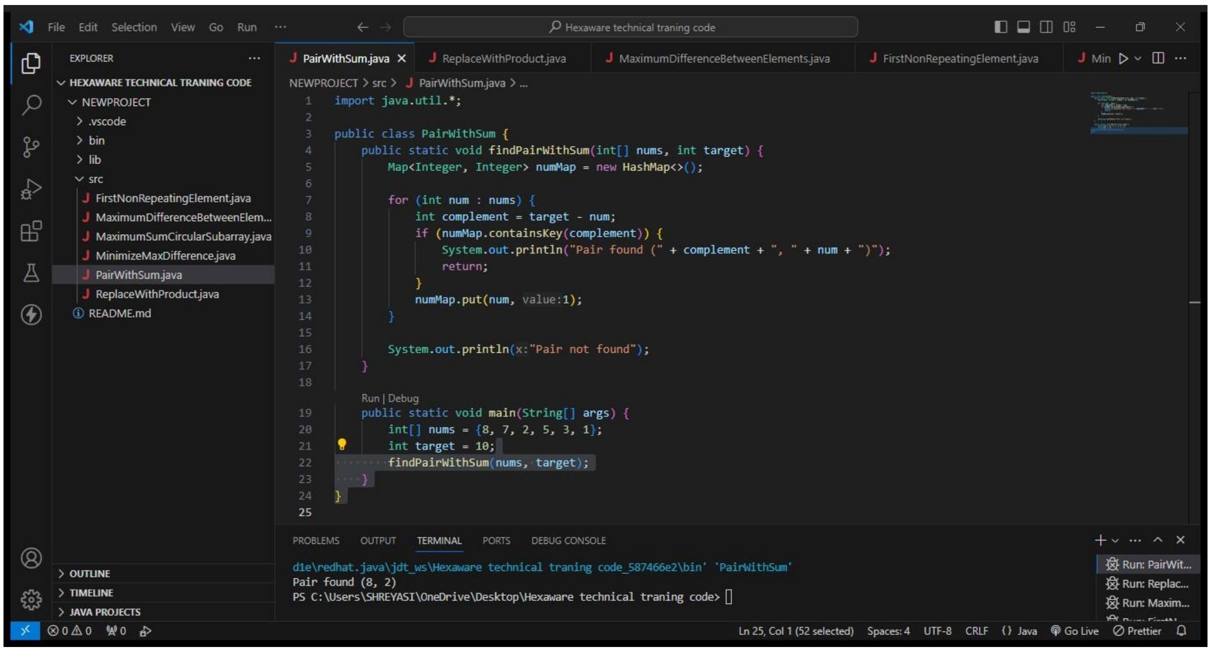
ANS:-

JAVA CODE:- import

```
java.util.*;


public class PairWithSum {
   public static void findPairWithSum(int[] nums, int target) {
      Map<Integer, Integer> numMap = new HashMap<>();


      for (int num : nums) {           int
complement = target - num;           if
(numMap.containsKey(complement)) {
            System.out.println("Pair found (" + complement + ",
" + num + ")");
```

```
            return;
          }
          numMap.put(num, 1);
       }

       System.out.println("Pair not found");
   }

   public static void main(String[] args) {
int[] nums = {8, 7, 2, 5, 3, 1};        int
target = 10;
findPairWithSum(nums, target);
   }
}
```

OUTPUT:- Pair found (8, 2)



```java
import java.util.*;

public class PairWithSum {
    public static void findPairWithSum(int[] nums, int target) {
        Map<Integer, Integer> numMap = new HashMap<>();

        for (int num : nums) {
            int complement = target - num;
            if (numMap.containsKey(complement)) {
                System.out.println("Pair found (" + complement + ", " + num + ")");
                return;
            }
            numMap.put(num, value:1);
        }

        System.out.println(x:"Pair not found");
    }

    public static void main(String[] args) {
        int[] nums = {8, 7, 2, 5, 3, 1};
        int target = 10;
        findPairWithSum(nums, target);
    }
}
```

```
d1e\redhat.java\jdt_ws\Hexaware technical traning code_587466e2\bin' 'PairWithSum'
Pair found (8, 2)
PS C:\Users\SHREYASI\OneDrive\Desktop\Hexaware technical traning code> 
```

Question-2

Given an integer array, replace each element with the product of every other element without using the division operator.

For example,

Input: { 1, 2, 3, 4, 5 }Output: { 120, 60, 40, 30, 24 } Input: { 5, 3, 4, 2, 6, 8 }Output: { 1152, 1920, 1440, 2880, 960, 720 }

ANS:-

JAVA CODE:- import

java.util.Arrays;

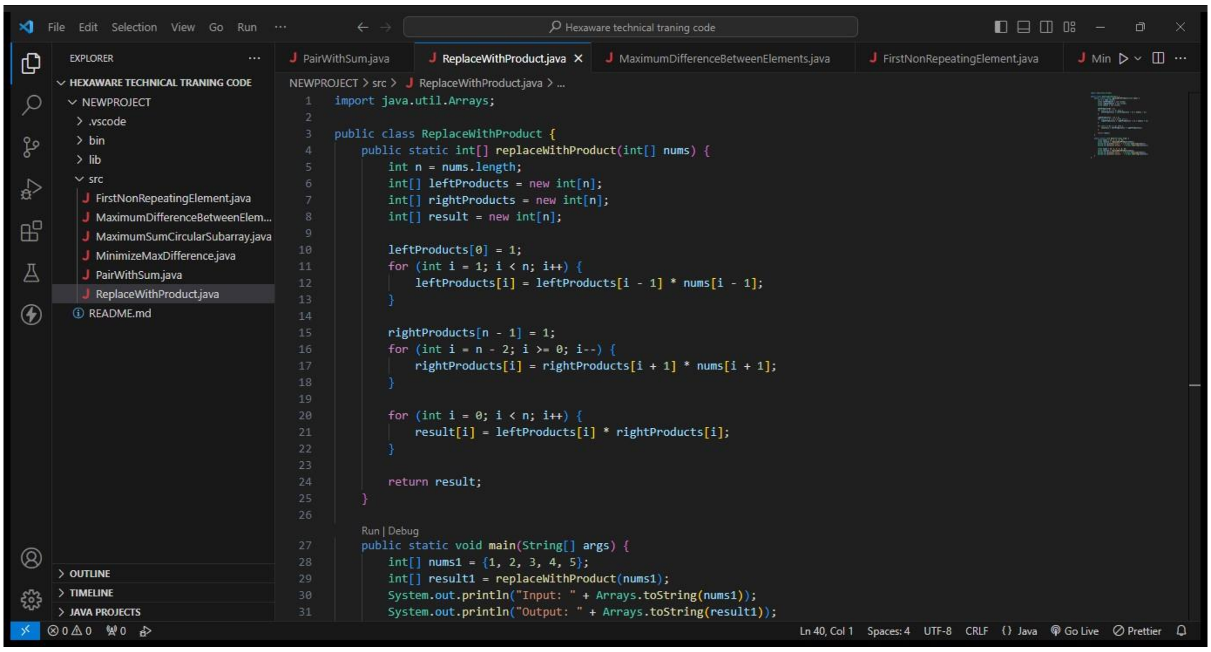
public class ReplaceWithProduct {

    public static int[] replaceWithProduct(int[] nums) {

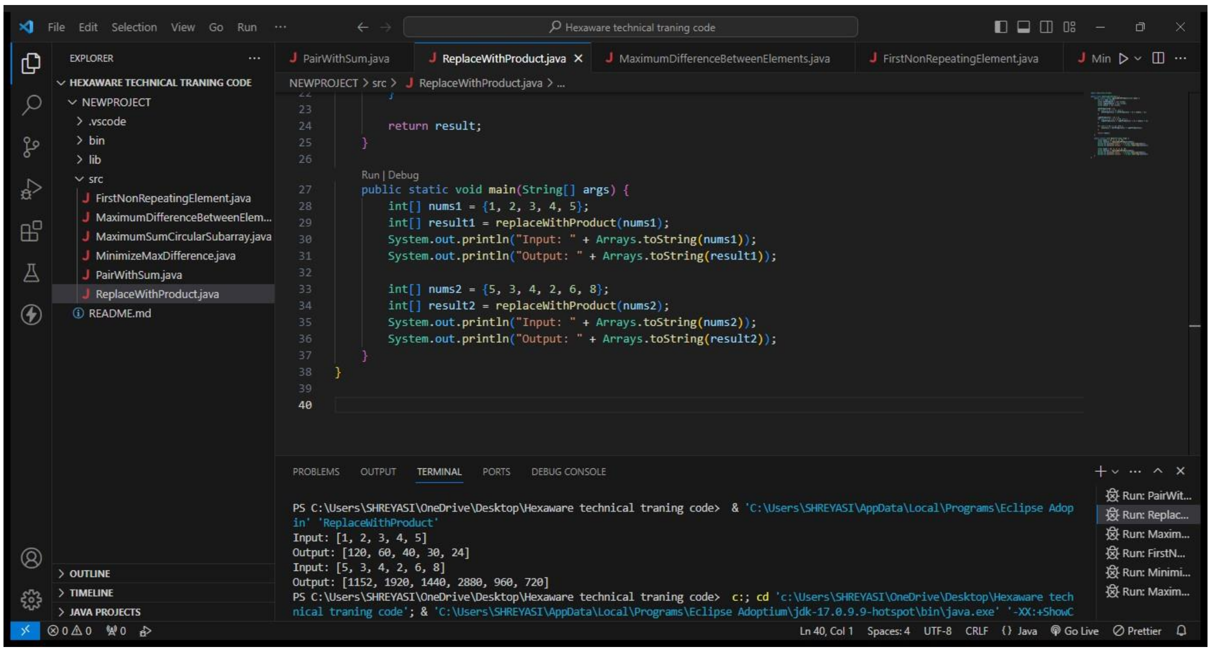
```
        int n = nums.length;         int[]
leftProducts = new int[n];         int[]
rightProducts = new int[n];         int[]
result = new int[n];

        leftProducts[0] = 1;
for (int i = 1; i < n; i++) {
            leftProducts[i] = leftProducts[i - 1] * nums[i - 1];
        }

        rightProducts[n - 1] = 1;
for (int i = n - 2; i >= 0; i--) {
            rightProducts[i] = rightProducts[i + 1] * nums[i + 1];
        }

        for (int i = 0; i < n; i++) {
            result[i] = leftProducts[i] * rightProducts[i];
        }

        return result;
```

```
    }


    public static void main(String[] args) {
int[] nums1 = {1, 2, 3, 4, 5};
        int[] result1 = replaceWithProduct(nums1);
        System.out.println("Input: " + Arrays.toString(nums1));
        System.out.println("Output: " + Arrays.toString(result1));

        int[] nums2 = {5, 3, 4, 2, 6, 8};
        int[] result2 = replaceWithProduct(nums2);
        System.out.println("Input: " + Arrays.toString(nums2));
        System.out.println("Output: " + Arrays.toString(result2));
    }
}
```

OUTPUT:-
```
Input: [1, 2, 3, 4, 5]
Output: [120, 60, 40, 30, 24]
Input: [5, 3, 4, 2, 6, 8]
Output: [1152, 1920, 1440, 2880, 960, 720]
```



```java
import java.util.Arrays;

public class ReplaceWithProduct {
    public static int[] replaceWithProduct(int[] nums) {
        int n = nums.length;
        int[] leftProducts = new int[n];
        int[] rightProducts = new int[n];
        int[] result = new int[n];

        leftProducts[0] = 1;
        for (int i = 1; i < n; i++) {
            leftProducts[i] = leftProducts[i - 1] * nums[i - 1];
        }

        rightProducts[n - 1] = 1;
        for (int i = n - 2; i >= 0; i--) {
            rightProducts[i] = rightProducts[i + 1] * nums[i + 1];
        }

        for (int i = 0; i < n; i++) {
            result[i] = leftProducts[i] * rightProducts[i];
        }

        return result;
    }

    public static void main(String[] args) {
        int[] nums1 = {1, 2, 3, 4, 5};
        int[] result1 = replaceWithProduct(nums1);
        System.out.println("Input: " + Arrays.toString(nums1));
        System.out.println("Output: " + Arrays.toString(result1));
```



```java

        return result;
    }

    public static void main(String[] args) {
        int[] nums1 = {1, 2, 3, 4, 5};
        int[] result1 = replaceWithProduct(nums1);
        System.out.println("Input: " + Arrays.toString(nums1));
        System.out.println("Output: " + Arrays.toString(result1));

        int[] nums2 = {5, 3, 4, 2, 6, 8};
        int[] result2 = replaceWithProduct(nums2);
        System.out.println("Input: " + Arrays.toString(nums2));
        System.out.println("Output: " + Arrays.toString(result2));
    }
}

```

```
PS C:\Users\SHREYASI\OneDrive\Desktop\Hexaware technical traning code>  & 'C:\Users\SHREYASI\AppData\Local\Programs\Eclipse Adop
in' 'ReplaceWithProduct'
Input: [1, 2, 3, 4, 5]
Output: [120, 60, 40, 30, 24]
Input: [5, 3, 4, 2, 6, 8]
Output: [1152, 1920, 1440, 2880, 960, 720]
PS C:\Users\SHREYASI\OneDrive\Desktop\Hexaware technical traning code>  c:; cd 'c:\Users\SHREYASI\OneDrive\Desktop\Hexaware tech
nical traning code'; & 'C:\Users\SHREYASI\AppData\Local\Programs\Eclipse Adoptium\jdk-17.0.9.9-hotspot\bin\java.exe' '-XX:+ShowC
```

Question-3

Maximum Sum Circular Subarray

Given a circular integer array, find a subarray with the largest sum in it.

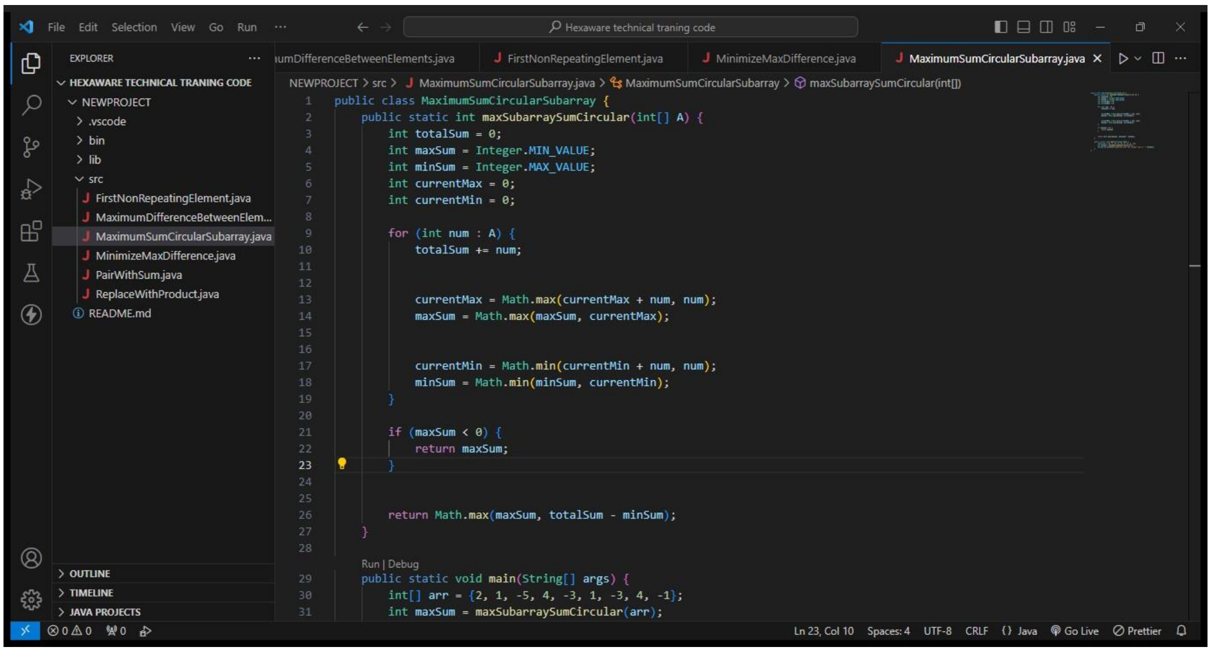
For example :Input: {2, 1, -5, 4, -3, 1, -3, 4, -1} Output: Subarray with the largest sum is {4, -1, 2, 1} with sum 6.

ANS: JAVA CODE: public class

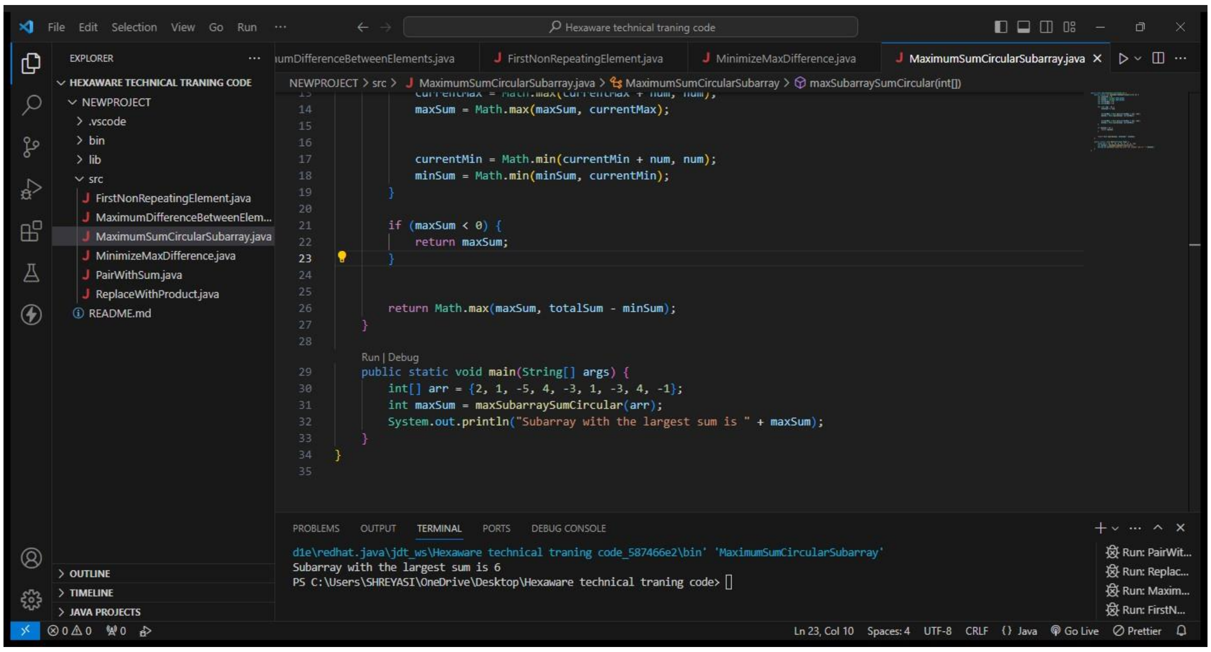
```
MaximumSumCircularSubarray {     public static int
maxSubarraySumCircular(int[] A) {         int totalSum
= 0;
       int maxSum = Integer.MIN_VALUE;
int minSum = Integer.MAX_VALUE;
int currentMax = 0;         int currentMin
= 0;

       for (int num : A) {
totalSum += num;


           currentMax = Math.max(currentMax + num, num);
maxSum = Math.max(maxSum, currentMax);


           currentMin = Math.min(currentMin + num, num);
minSum = Math.min(minSum, currentMin);
```

```
        }


        if (maxSum < 0) {
return maxSum;
        }



        return Math.max(maxSum, totalSum - minSum);
    }


    public static void main(String[] args) {
int[] arr = {2, 1, -5, 4, -3, 1, -3, 4, -1};
        int maxSum = maxSubarraySumCircular(arr);
        System.out.println("Subarray with the largest sum is " +
maxSum);
    }
}
```

OUTPUT: Subarray with the largest sum is 6



```java
public class MaximumSumCircularSubarray {
    public static int maxSubarraySumCircular(int[] A) {
        int totalSum = 0;
        int maxSum = Integer.MIN_VALUE;
        int minSum = Integer.MAX_VALUE;
        int currentMax = 0;
        int currentMin = 0;

        for (int num : A) {
            totalSum += num;


            currentMax = Math.max(currentMax + num, num);
            maxSum = Math.max(maxSum, currentMax);


            currentMin = Math.min(currentMin + num, num);
            minSum = Math.min(minSum, currentMin);
        }

        if (maxSum < 0) {
            return maxSum;
        }


        return Math.max(maxSum, totalSum - minSum);
    }

    public static void main(String[] args) {
        int[] arr = {2, 1, -5, 4, -3, 1, -3, 4, -1};
        int maxSum = maxSubarraySumCircular(arr);
```



```java
            maxSum = Math.max(maxSum, currentMax);


            currentMin = Math.min(currentMin + num, num);
            minSum = Math.min(minSum, currentMin);
        }

        if (maxSum < 0) {
            return maxSum;
        }


        return Math.max(maxSum, totalSum - minSum);
    }

    public static void main(String[] args) {
        int[] arr = {2, 1, -5, 4, -3, 1, -3, 4, -1};
        int maxSum = maxSubarraySumCircular(arr);
        System.out.println("Subarray with the largest sum is " + maxSum);
    }
}
```

```
d1e\redhat.java\jdt_ws\Hexaware technical traning code_587466e2\bin' 'MaximumSumCircularSubarray'
Subarray with the largest sum is 6
PS C:\Users\SHREYASI\OneDrive\Desktop\Hexaware technical traning code>
```

Question-4:

Find the maximum difference between two array elements that satisfies the given constraints

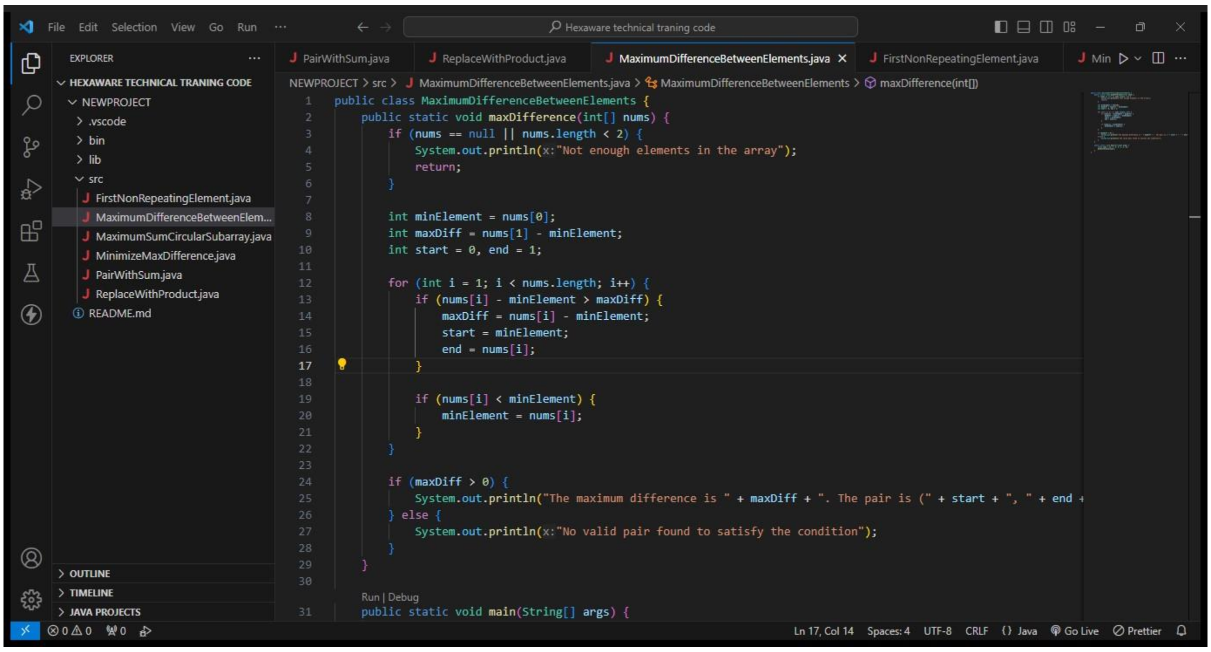
Given an integer array, find the maximum difference between two elements in it such that the smaller element appears before the larger element.

For example:Input: { 2, 7, 9, 5, 1, 3, 5 } Output: The maximum difference is 7. The pair is (2, 9) ANS: JAVA CODE:

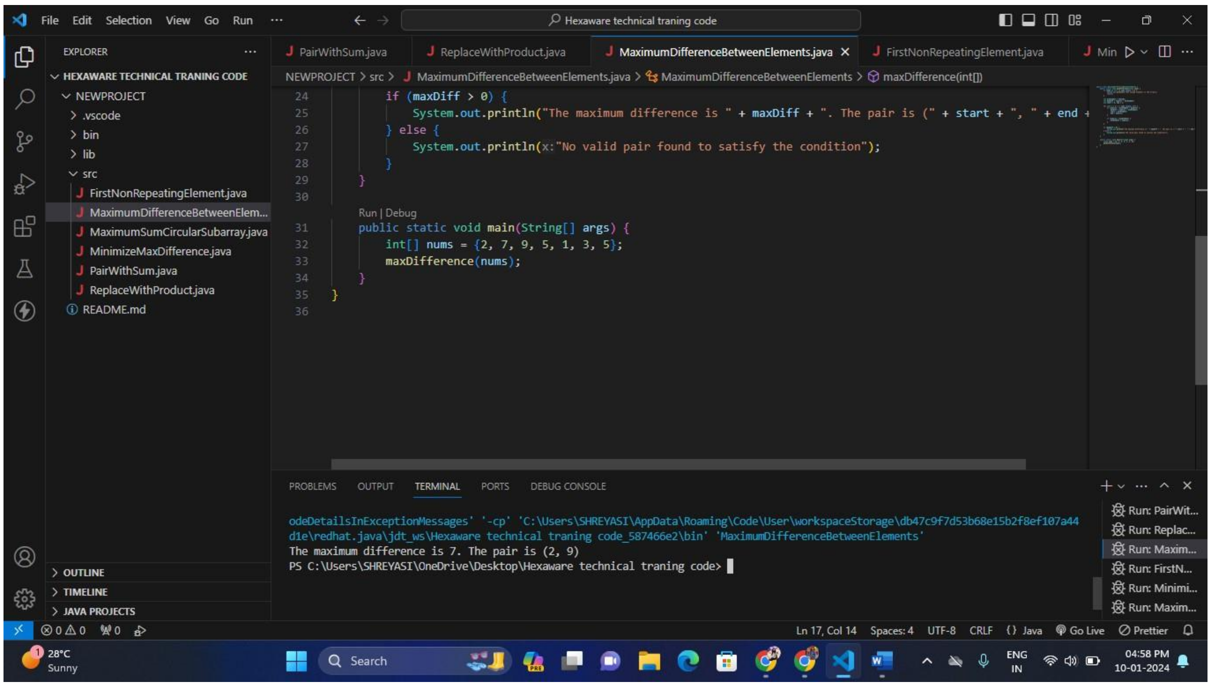
```
public class
MaximumDifferenceBetweenElements {     public static void
maxDifference(int[] nums) {         if (nums == null ||
nums.length < 2) {
            System.out.println("Not enough elements in the
array");
            return;
        }

        int minElement = nums[0];          int
maxDiff = nums[1] - minElement;
int start = 0, end = 1;

        for (int i = 1; i < nums.length; i++) {
if (nums[i] - minElement > maxDiff) {
maxDiff = nums[i] - minElement;
start = minElement;
                end = nums[i];
            }
        }

            if (nums[i] < minElement) {
minElement = nums[i];
            }
        }

        if (maxDiff > 0) {
            System.out.println("The maximum difference is " +
maxDiff + ". The pair is (" + start + ", " + end + ")");
        } else {
            System.out.println("No valid pair found to satisfy the
condition");
        }
    }

    public static void main(String[] args) {
int[] nums = {2, 7, 9, 5, 1, 3, 5};
maxDifference(nums);

    }
}
```

OUTPUT:

The maximum difference is 7. The pair is (2, 9)



```java
public class MaximumDifferenceBetweenElements {
    public static void maxDifference(int[] nums) {
        if (nums == null || nums.length < 2) {
            System.out.println(x:"Not enough elements in the array");
            return;
        }

        int minElement = nums[0];
        int maxDiff = nums[1] - minElement;
        int start = 0, end = 1;

        for (int i = 1; i < nums.length; i++) {
            if (nums[i] - minElement > maxDiff) {
                maxDiff = nums[i] - minElement;
                start = minElement;
                end = nums[i];
            }

            if (nums[i] < minElement) {
                minElement = nums[i];
            }
        }

        if (maxDiff > 0) {
            System.out.println("The maximum difference is " + maxDiff + ". The pair is (" + start + ", " + end +
        } else {
            System.out.println(x:"No valid pair found to satisfy the condition");
        }
    }

    Run | Debug
    public static void main(String[] args) {
```



```java
        if (maxDiff > 0) {
            System.out.println("The maximum difference is " + maxDiff + ". The pair is (" + start + ", " + end +
        } else {
            System.out.println(x:"No valid pair found to satisfy the condition");
        }
    }

    Run | Debug
    public static void main(String[] args) {
        int[] nums = {2, 7, 9, 5, 1, 3, 5};
        maxDifference(nums);
    }
}
```

```
odeDetailsInExceptionMessages' '-cp' 'C:\Users\SHREYASI\AppData\Roaming\Code\User\workspaceStorage\db47c9f7d53b68e15b2f8ef107a44
d1e\redhat.java\jdt_ws\Hexaware technical traning code_587466e2\bin' 'MaximumDifferenceBetweenElements'
The maximum difference is 7. The pair is (2, 9)
PS C:\Users\SHREYASI\OneDrive\Desktop\Hexaware technical traning code>
```

Question:5

Given an array of integers of size N, the task is to find the first non-repeating element in this array.

Examples:

Input: {-1, 2, -1, 3, 0}

Output: 2

Explanation: The first number that does not repeat is : 2

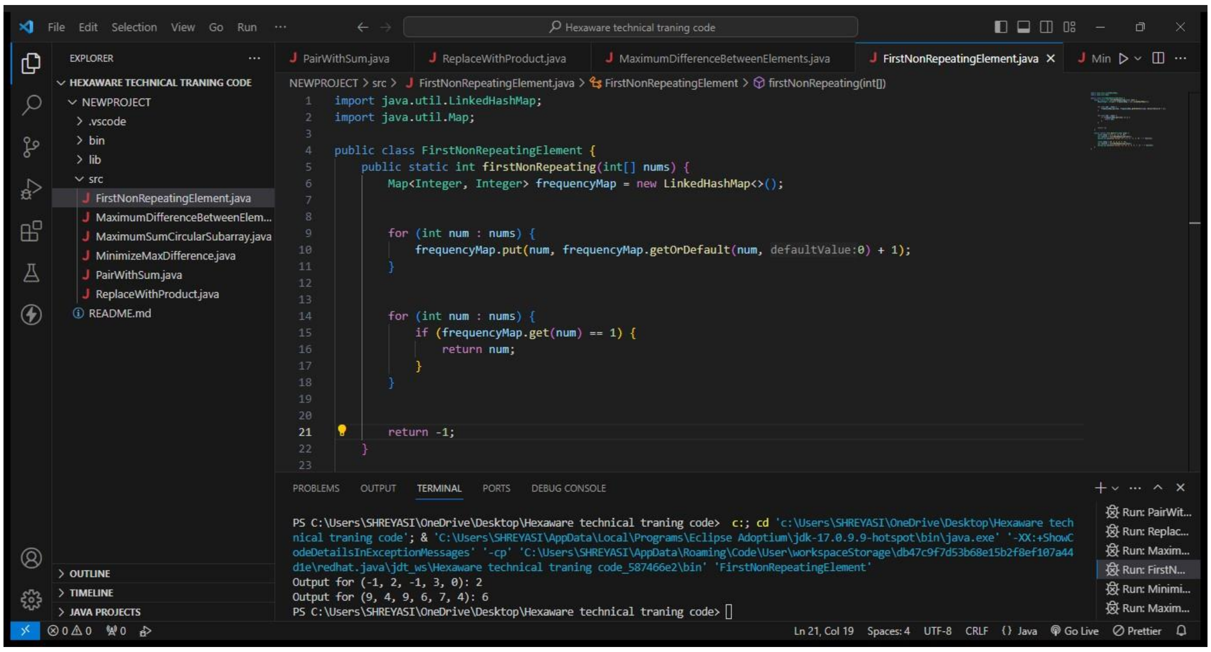
Input: {9, 4, 9, 6, 7, 4}

ANS: JAVA CODE:
```
import
java.util.LinkedHashMap; import
java.util.Map;


public class FirstNonRepeatingElement {     public
static int firstNonRepeating(int[] nums) {
        Map<Integer, Integer> frequencyMap = new
LinkedHashMap<>();



        for (int num : nums) {
            frequencyMap.put(num,
frequencyMap.getOrDefault(num, 0) + 1);
        }
        for (int num : nums) {
            if (frequencyMap.get(num) == 1) {
return num;
            }
        }



        return -1;
    }

    public static void main(String[] args) {
int[] nums1 = {-1, 2, -1, 3, 0};         int result1
= firstNonRepeating(nums1);
        System.out.println("Output for (-1, 2, -1, 3, 0): " +
result1);

        int[] nums2 = {9, 4, 9, 6, 7, 4};         int
result2 = firstNonRepeating(nums2);
```

```
        System.out.println("Output for (9, 4, 9, 6, 7, 4): " +
result2);
    }
}
```

OUTPUT:

```
Output for (-1, 2, -1, 3, 0): 2

Output for (9, 4, 9, 6, 7, 4): 6
```



```java
import java.util.LinkedHashMap;
import java.util.Map;

public class FirstNonRepeatingElement {
    public static int firstNonRepeating(int[] nums) {
        Map<Integer, Integer> frequencyMap = new LinkedHashMap<>();


        for (int num : nums) {
            frequencyMap.put(num, frequencyMap.getOrDefault(num, defaultValue:0) + 1);
        }


        for (int num : nums) {
            if (frequencyMap.get(num) == 1) {
                return num;
            }
        }


        return -1;
    }

```

```
PS C:\Users\SHREYASI\OneDrive\Desktop\Hexaware technical traning code>  c:; cd 'c:\Users\SHREYASI\OneDrive\Desktop\Hexaware tech
nical traning code'; & 'C:\Users\SHREYASI\AppData\Local\Programs\Eclipse Adoptium\jdk-17.0.9.9-hotspot\bin\java.exe' '-XX:+ShowC
odeDetailsInExceptionMessages' '-cp' 'C:\Users\SHREYASI\AppData\Roaming\Code\User\workspaceStorage\db47c9f7d53b68e15b2f8ef107a44
d1e\redhat.java\jdt_ws\Hexaware technical traning code_587466e2\bin' 'FirstNonRepeatingElement'
Output for (-1, 2, -1, 3, 0): 2
Output for (9, 4, 9, 6, 7, 4): 6
PS C:\Users\SHREYASI\OneDrive\Desktop\Hexaware technical traning code> 
```

Question:6

Minimize the maximum difference between the heights

Given the heights of N towers and a value of K, Either increase or decrease the height of every tower by K (only once) where K > 0. After modifications, the task is to minimize the difference between the heights of the longest and the shortest tower and output its difference.

Examples:

Input: arr[] = {1, 15, 10}, k = 6

Output: Maximum difference is 5.

Explanation: Change 1 to 7, 15 to 9 and 10 to 4. Maximum difference is 5 (between 4 and 9). We can’t get a lower difference.

Input: arr[] = {1, 5, 15, 10}, k = 3

Output: Maximum difference is 8, arr[] = {4, 8, 12, 7}

ANS:

JAVA CODE:

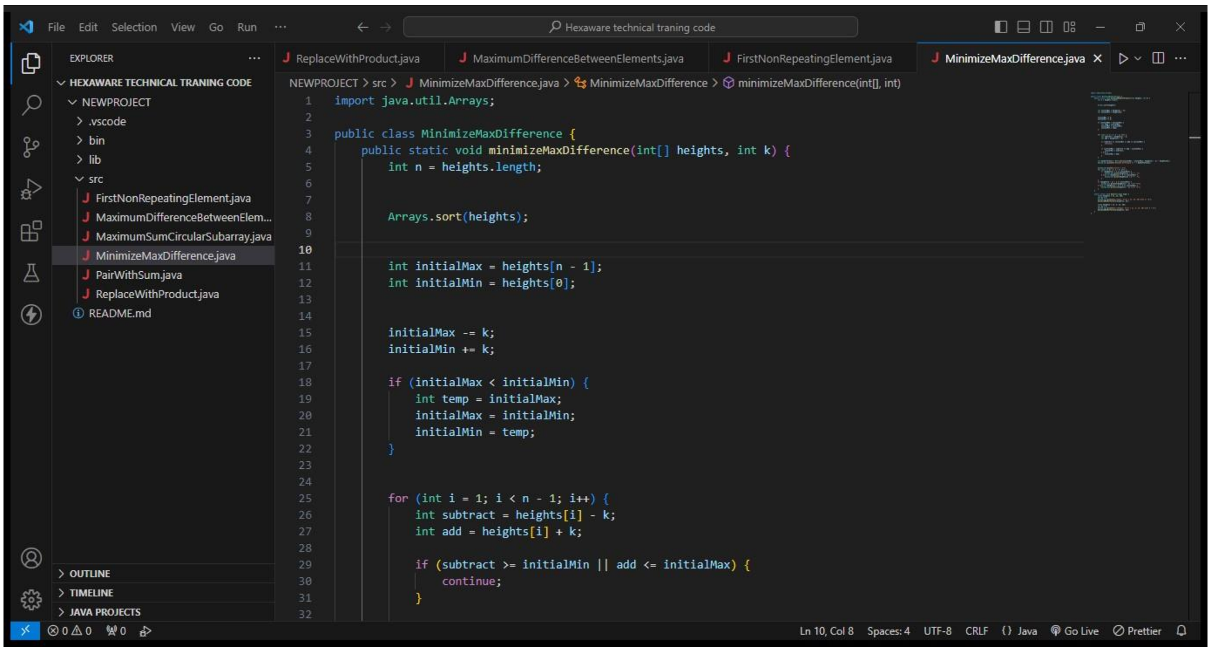
```
import java.util.Arrays;


public class MinimizeMaxDifference {

   public static void minimizeMaxDifference(int[] heights, int
k) {

       int n = heights.length;



       Arrays.sort(heights);
```

```
        int initialMax = heights[n - 1];
int initialMin = heights[0];



        initialMax -= k;
initialMin += k;

        if (initialMax < initialMin) {
int temp = initialMax;
initialMax = initialMin;
initialMin = temp;
        }

        for (int i = 1; i < n - 1; i++) {
int subtract = heights[i] - k;
int add = heights[i] + k;

            if (subtract >= initialMin || add <= initialMax) {
continue;
```

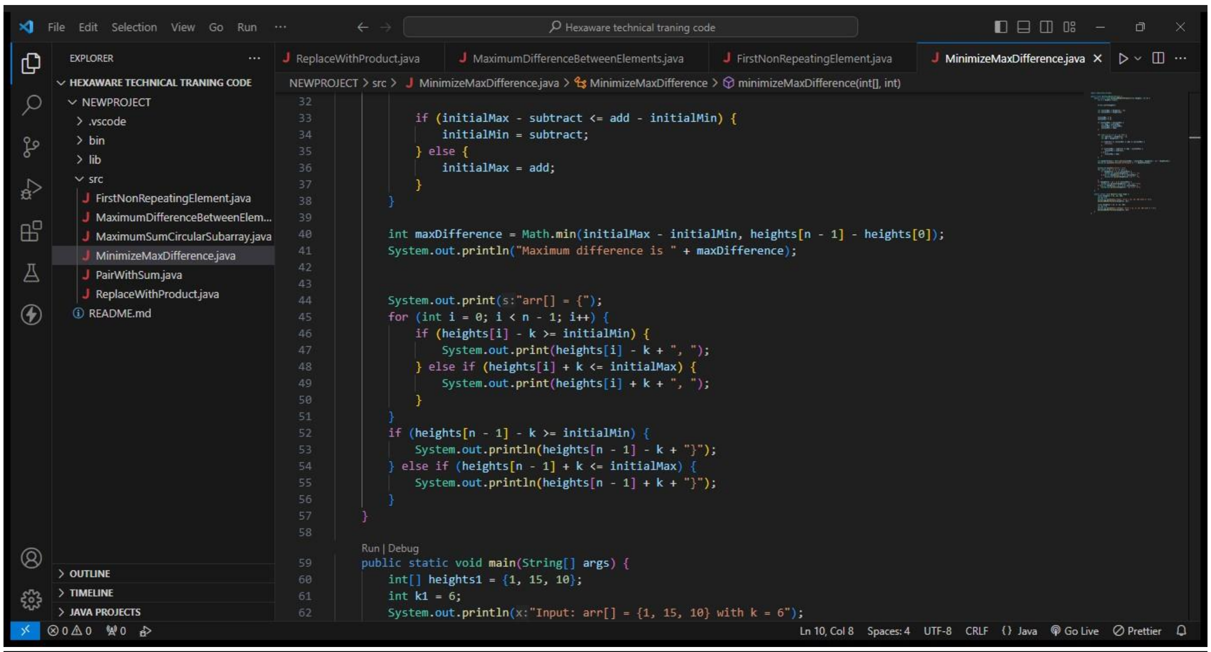
```
            }

            if (initialMax - subtract <= add - initialMin) {
initialMin = subtract;
            } else {
                initialMax = add;
            }
        }

        int maxDifference = Math.min(initialMax - initialMin,
heights[n - 1] - heights[0]);
        System.out.println("Maximum difference is " +
maxDifference);


        System.out.print("arr[] = {");
for (int i = 0; i < n - 1; i++) {             if
(heights[i] - k >= initialMin) {
                System.out.print(heights[i] - k + ", ");
            } else if (heights[i] + k <= initialMax) {
                System.out.print(heights[i] + k + ", ");
            }
        }
        if (heights[n - 1] - k >= initialMin) {
            System.out.println(heights[n - 1] - k + "}");
        } else if (heights[n - 1] + k <= initialMax) {
            System.out.println(heights[n - 1] + k + "}");
        }
    }

    public static void main(String[] args) {
int[] heights1 = {1, 15, 10};        int k1 =
6;
        System.out.println("Input: arr[] = {1, 15, 10} with k = 6");
minimizeMaxDifference(heights1, k1);


        int[] heights2 = {1, 5, 15, 10};
int k2 = 3;
        System.out.println("\nInput: arr[] = {1, 5, 15, 10} with k =
3");
        minimizeMaxDifference(heights2, k2);
    }
}
```

OUTPUT:

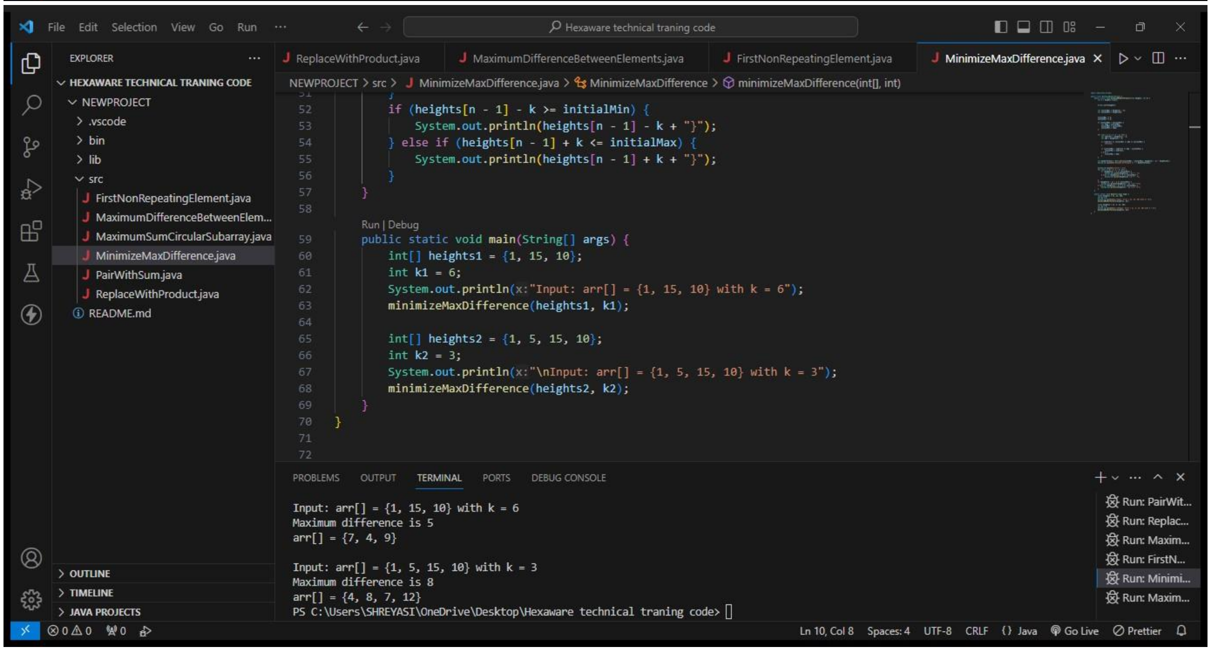
```
Input: arr[] = {1, 15, 10} with k = 6 Maximum
difference is 5
arr[] = {7, 4, 9}


Input: arr[] = {1, 5, 15, 10} with k = 3 Maximum
difference is 8
arr[] = {4, 8, 7, 12}
```



```java
import java.util.Arrays;

public class MinimizeMaxDifference {
    public static void minimizeMaxDifference(int[] heights, int k) {
        int n = heights.length;


        Arrays.sort(heights);


        int initialMax = heights[n - 1];
        int initialMin = heights[0];


        initialMax -= k;
        initialMin += k;

        if (initialMax < initialMin) {
            int temp = initialMax;
            initialMax = initialMin;
            initialMin = temp;
        }


        for (int i = 1; i < n - 1; i++) {
            int subtract = heights[i] - k;
            int add = heights[i] + k;

            if (subtract >= initialMin || add <= initialMax) {
                continue;
            }

```


File Edit Selection View Go Run
Hexaware technical traning code
EXPLORER
HEXAWARE TECHNICAL TRANING CODE
NEWPROJECT
.vscode
bin
lib
src
FirstNonRepeatingElement.java
MaximumDifferenceBetweenElem...
MaximumSumCircularSubarray.java
MinimizeMaxDifference.java
PairWithSum.java
ReplaceWithProduct.java
README.md
OUTLINE
TIMELINE
JAVA PROJECTS
ReplaceWithProduct.java
MaximumDifferenceBetweenElements.java
FirstNonRepeatingElement.java
MinimizeMaxDifference.java
NEWPROJECT > src > MinimizeMaxDifference.java > MinimizeMaxDifference > minimizeMaxDifference(int[], int)
Ln 10, Col 8 Spaces: 4 UTF-8 CRLF Java Go Live Prettier

```

                if (initialMax - subtract <= add - initialMin) {
                    initialMin = subtract;
                } else {
                    initialMax = add;
                }
            }

            int maxDifference = Math.min(initialMax - initialMin, heights[n - 1] - heights[0]);
            System.out.println("Maximum difference is " + maxDifference);


            System.out.print(s:"arr[] = {");
            for (int i = 0; i < n - 1; i++) {
                if (heights[i] - k >= initialMin) {
                    System.out.print(heights[i] - k + ", ");
                } else if (heights[i] + k <= initialMax) {
                    System.out.print(heights[i] + k + ", ");
                }
            }
            if (heights[n - 1] - k >= initialMin) {
                System.out.println(heights[n - 1] - k + "}");
            } else if (heights[n - 1] + k <= initialMax) {
                System.out.println(heights[n - 1] + k + "}");
            }
        }

        Run | Debug
        public static void main(String[] args) {
            int[] heights1 = {1, 15, 10};
            int k1 = 6;
            System.out.println(x:"Input: arr[] = {1, 15, 10} with k = 6");
```


File Edit Selection View Go Run
Hexaware technical traning code
EXPLORER
HEXAWARE TECHNICAL TRANING CODE
NEWPROJECT
.vscode
bin
lib
src
FirstNonRepeatingElement.java
MaximumDifferenceBetweenElem...
MaximumSumCircularSubarray.java
MinimizeMaxDifference.java
PairWithSum.java
ReplaceWithProduct.java
README.md
OUTLINE
TIMELINE
JAVA PROJECTS
NEWPROJECT > src > MinimizeMaxDifference.java > MinimizeMaxDifference > minimizeMaxDifference(int[], int)
PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE
Run: PairWit...
Run: Replac...
Run: Maxim...
Run: FirstN...
Run: Minimi...
Run: Maxim...
Ln 10, Col 8 Spaces: 4 UTF-8 CRLF Java Go Live Prettier

```
            if (heights[n - 1] - k >= initialMin) {
                System.out.println(heights[n - 1] - k + "}");
            } else if (heights[n - 1] + k <= initialMax) {
                System.out.println(heights[n - 1] + k + "}");
            }
        }

        Run | Debug
        public static void main(String[] args) {
            int[] heights1 = {1, 15, 10};
            int k1 = 6;
            System.out.println(x:"Input: arr[] = {1, 15, 10} with k = 6");
            minimizeMaxDifference(heights1, k1);

            int[] heights2 = {1, 5, 15, 10};
            int k2 = 3;
            System.out.println(x:"\nInput: arr[] = {1, 5, 15, 10} with k = 3");
            minimizeMaxDifference(heights2, k2);
        }
    }


```

```
Input: arr[] = {1, 15, 10} with k = 6
Maximum difference is 5
arr[] = {7, 4, 9}

Input: arr[] = {1, 5, 15, 10} with k = 3
Maximum difference is 8
arr[] = {4, 8, 7, 12}
PS C:\Users\SHREYASI\OneDrive\Desktop\Hexaware technical traning code>
```