Insertion Sort

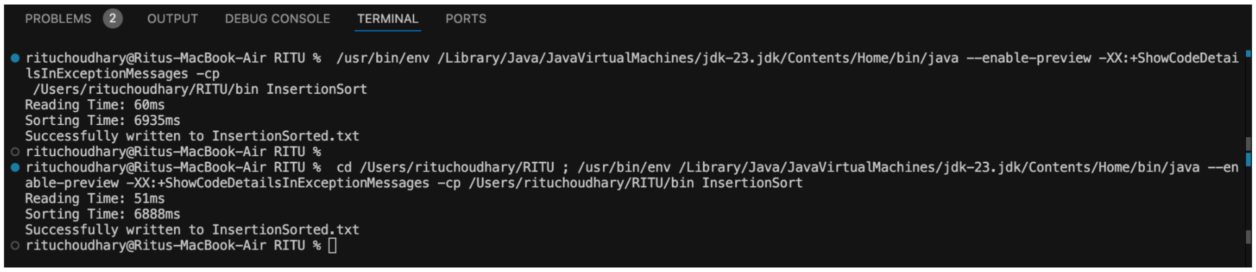
Name : Ritu Choudhary Roll no : 51 Batch : b3

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

class InsertionSort {
   public static void main(String[] args) {
       String inputPath = "file2.txt";
       String outputPath = "InsertionSort.txt";
       ArrayList<Integer> numbers = new ArrayList<>();
       long startTime = System.currentTimeMillis();
       try (BufferedReader br = new BufferedReader(new FileReader(inputPath))) {
           String line;
           while ((line = br.readLine()) != null) {
               String[] values = line.split("\\s+");
               for (String value : values) {
                   numbers.add(Integer.parseInt(value));
               }
           }
       } catch (IOException e) {
           System.out.println("Error reading file: " + e.getMessage());
           return;
       }
       long endTime = System.currentTimeMillis();
       long timeReq = (endTime - startTime);
       System.out.println("Reading Time: " + timeReq + "ms");
       startTime = System.currentTimeMillis();
       insertionSort(numbers);
       endTime = System.currentTimeMillis();
       timeReq = (endTime - startTime);
       System.out.println("Sorting Time: " + timeReq + "ms");
       try (PrintWriter pw = new PrintWriter(new FileWriter(outputPath))) {
           for (Integer number : numbers) {
               pw.print(number + "\t");
           }
           System.out.println("Successfully written to InsertionSorted.txt");
       } catch (IOException e) {
           System.out.println("Error writing file: " + e.getMessage());
       }
   }
```

```
    private static void insertionSort(ArrayList<Integer> arr) {
        for (int i = 1; i < arr.size(); i++) {
            int key = arr.get(i);
            int j = i - 1;
            while (j >= 0 && arr.get(j) > key) {
                arr.set(j + 1, arr.get(j));
                j = j - 1;
            }
            arr.set(j + 1, key);
        }
    }
}
```

Output :



PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
● rituchoudhary@Ritus-MacBook-Air RITU %  /usr/bin/env /Library/Java/JavaVirtualMachines/jdk-23.jdk/Contents/Home/bin/java --enable-preview -XX:+ShowCodeDetai
  lsInExceptionMessages -cp
   /Users/rituchoudhary/RITU/bin InsertionSort
  Reading Time: 60ms
  Sorting Time: 6935ms
  Successfully written to InsertionSorted.txt
○ rituchoudhary@Ritus-MacBook-Air RITU %
● rituchoudhary@Ritus-MacBook-Air RITU %  cd /Users/rituchoudhary/RITU ; /usr/bin/env /Library/Java/JavaVirtualMachines/jdk-23.jdk/Contents/Home/bin/java --en
  able-preview -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/rituchoudhary/RITU/bin InsertionSort
  Reading Time: 51ms
  Sorting Time: 6888ms
  Successfully written to InsertionSorted.txt
○ rituchoudhary@Ritus-MacBook-Air RITU % 
```

Comparing With SelectionSort

Insertion Sort is faster for small or nearly sorted datasets due to its adaptive nature, with a best-case time complexity of O(n).

Selection Sort always performs O(n²) comparisons, making it generally slower than Insertion Sort for most inputs, but it requires fewer swaps.

Insertion Sort is preferred for small or partially sorted arrays, while Selection Sort is simpler but less efficient in practice.