**1. You are working for an e-commerce company that deals with a large volume of customer orders. The company wants to optimize its warehouse operations by analyzing the efficiency of order processing.**

**The goal is to count the number of inversion pairs in the order processing times to identify potential inefficiencies. An inversion pair in this context is defined as a situation where a later order (in terms of processing sequence) took less time to process than an earlier order. By identifying these inversions, you can pinpoint areas where the warehouse team may need to improve their workflow or processes.**

**Input**

**An array P[] of n integers representing the processing times of orders, in the order they were processed.**

**Output**

**The count of inversion pairs in the array.**

**CODE:**

```
public class CountInversions {
private static int mergeAndCount(int[] arr, int[] tempArr, int left, int mid, int right) {
int i = left;
int j = mid + 1;
int k = left;
int invCount = 0;

while (i <= mid && j <= right) {
if (arr[i] <= arr[j]) {
tempArr[k++] = arr[i++];
} else {
tempArr[k++] = arr[j++];
invCount += (mid - i + 1);
}
}
```

```
while (i <= mid) {
tempArr[k++] = arr[i++];
}

while (j <= right) {
tempArr[k++] = arr[j++];
}

System.arraycopy(tempArr, left, arr, left, right - left + 1);

return invCount;
}

private static int mergeSortAndCount(int[] arr, int[] tempArr, int left, int right) {
int invCount = 0;
if (left < right) {
int mid = (left + right) / 2;
invCount += mergeSortAndCount(arr, tempArr, left, mid);
invCount += mergeSortAndCount(arr, tempArr, mid + 1, right);
invCount += mergeAndCount(arr, tempArr, left, mid, right);
}
return invCount;
}

public static int countInversions(int[] P) {
int n = P.length;
int[] tempArr = new int[n];
return mergeSortAndCount(P, tempArr, 0, n - 1);
```

```
}

public static void main(String[] args) {
int[] P = {5, 3, 2, 4, 1};
System.out.println("Number of inversion pairs: " + countInversions(P));
}
}
```

**OUTPUT:**

RunNumber of inversion pairs: 8

**2. You work as a software engineer for a company that manages online retail stores. The company wants to implement a feature that provides real-time insights into daily sales trends. One of the key metrics is the cumulative sales amount up to a specific hour of the day.**

**The goal is to compute the prefix sum of the sales array, where each element in the array represents the sales amount at a specific hour. The prefix sum at any given hour is the total sales amount from the start of the day up to that hour.**

**Input**

**An array Sales[] of n integers representing the sales amount at each**

**hour of the day. Output**

**An array PrefixSum[] of n integers where PrefixSum[i] is the cumulative sales amount from the start of the day up to hour i.**

**CODE:**

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

```
public class PrefixSumCalculator {
    public static int[] computePrefixSum(int[] sales) {
        int n = sales.length;
        int[] prefixSum = new int[n];

        if (n > 0) {
            prefixSum[0] = sales[0];
            for (int i = 1; i < n; i++) {
                prefixSum[i] = prefixSum[i - 1] + sales[i];
            }
        }

        return prefixSum;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the number of hours:");
        int n = scanner.nextInt();
        int[] sales = new int[n];
        System.out.println("Enter the sales data:");
        for (int i = 0; i < n; i++) {
            sales[i] = scanner.nextInt();
        }

        int[] prefixSum = computePrefixSum(sales);
        System.out.println("Prefix Sum Array: " + Arrays.toString(prefixSum));

        scanner.close();
```

```
    }
}
```

**OUTPUT:**

Enter the number of hours:

5

Enter the sales data:

10

20

15

30

25

Prefix Sum Array: [10, 30, 45, 75, 100]

**3. You are a data analyst for a logistics company that tracks the fuel consumption of delivery vehicles throughout their routes. The company wants to optimize fuel usage by analyzing the remaining fuel capacity after each delivery point.**

**The goal is to compute the postfix sum of the fuel consumption array, where each element in the array represents the fuel consumed at a specific delivery point. The postfix sum at any given point is the total fuel consumption from that point to the end of the route.**

**Input**

**An array Fuel[] of m integers representing the fuel consumed at each**

**delivery point. Output**

**An array PostfixSum[] of m integers where PostfixSum[i] is the total fuel consumption from delivery point i to the end of the route.**

**CODE:**

```
import java.util.Arrays;
public class PostFixSum {
   public static int[] computePostfixSum(int[] fuel) {
```

```
        int m = fuel.length;
        int[] postfixSum = new int[m];
        // Start from the last element
        postfixSum[m - 1] = fuel[m - 1];
        // Compute the postfix sum from right to left
        for (int i = m - 2; i >= 0; i--) {
            postfixSum[i] = fuel[i] + postfixSum[i + 1];
        }
        return postfixSum;
    }
    public static void main(String[] args) {
        int[] fuel = {5, 10, 3, 7, 8};
        int[] postfixSum = computePostfixSum(fuel);
        System.out.println("PostfixSum[] = " + Arrays.toString(postfixSum));
    }
}
```

**OUTPUT:**

PostfixSum[] = [33, 28, 18, 15, 8]

**4. You work for a mobile app development company that specializes in creating educational apps for kids. One of your tasks is to develop a game that helps children learn and recognize different digits.**

**The game includes a feature where a list of numbers is provided, and the child needs to identify all the distinct digits present in that list. The goal is to find all unique digits from the given array of integers.**

**Input**

**An array N[] of k integers.**

**Output**

**A sorted list of distinct digits present in the array.**

**CODE:**

```
import java.util.Set;
import java.util.TreeSet;

public class UniqueDigits {
 public static void main(String[] args) {
 int[] N = {111, 222, 333, 4444, 666};
 System.out.println(findUniqueDigits(N));
 }

 public static Set<Integer> findUniqueDigits(int[] numbers) {
 Set<Integer> uniqueDigits = new TreeSet<>(); // TreeSet to store sorted
unique digits
 for (int num : numbers) {
 while (num > 0) {
 uniqueDigits.add(num % 10); // Extract digit
 num /= 10; // Remove last digit
 }
 }

 return uniqueDigits;
 }
}
```

**OUTPUT:**

[1, 2, 3, 4, 6]