



CSE3032 - Competitive Programming

WIN SEM (2022-2023) AMR

Class Number: AP2022236001007

Slot: L11+L12+L19+L20

ASSIGNMENT - 12

Last Date for Submission: Thursday (04-05-2023) @ 12.40PM

Name: Chilamakuru Sai Venkata Yaswanth

Reg No: 20BCI7076

Write the program using (C / C++ / Java / Python) to solve the following problems.

Concept: Cycle Sort & Merge Intervals

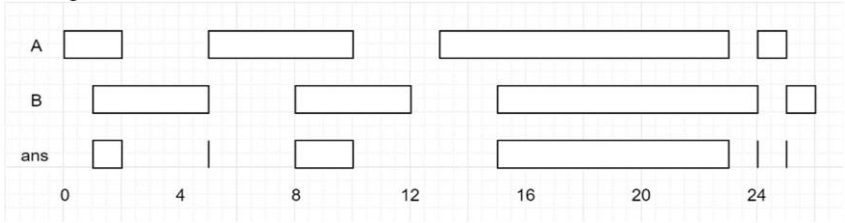
S.No	Problem Name	Statement
1	Find All Duplicates in an Array	<p>Given an integer array <code>nums</code> of length <code>n</code> where all the integers of <code>nums</code> are in the range <code>[1, n]</code> and each integer appears once or twice, return an array of all the integers that appears twice.</p> <p>You must write an algorithm that runs in $O(n)$ time and uses only constant extra space.</p> <p>Input: <code>nums = [4,3,2,7,8,2,3,1]</code> Output: <code>[2,3]</code></p> <p>Input: <code>nums = [1,1,2]</code> Output: <code>[1]</code></p> <p>Input: <code>nums = [1]</code> Output: <code>[]</code></p>
2	Find All Numbers Disappeared in an Array Or Find all Missing Numbers	<p>Given an array <code>nums</code> of <code>n</code> integers where <code>nums[i]</code> is in the range <code>[1, n]</code>, return an array of all the integers in the range <code>[1, n]</code> that do not appear in <code>nums</code>.</p> <p>Input: <code>nums = [4,3,2,7,8,2,3,1]</code> Output: <code>[5,6]</code></p> <p>Input: <code>nums = [1,1]</code> Output: <code>[2]</code></p>

3	Set Mismatch Or Find the Corrupt Pair	<p>You have a set of integers s, which originally contains all the numbers from 1 to n. Unfortunately, due to some error, one of the numbers in s got duplicated to another number in the set, which results in repetition of one number and loss of another number.</p> <p>You are given an integer array nums representing the data status of this set after the error.</p> <p>Find the number that occurs twice and the number that is missing and return them in the form of an array.</p> <p>Input: nums = [1,2,2,4]</p> <p>Output: [2,3]</p> <p>Input: nums = [1,1]</p> <p>Output: [1,2]</p>
4	Find the Duplicate Number	<p>Given an unsorted array containing n+1 numbers taken from the range 1 to n. The array has only one duplicate but it can be repeated multiple times. Find that duplicate number without using any extra space. You are, however, allowed to modify the input array.</p>

		<p>[1, 4, 4, 3, 2] □ 4</p> <p>[2, 1, 3, 3, 5, 4] □ 3</p> <p>[2, 4, 1, 4, 4] □ 4</p>
5	Find all Duplicate Numbers	<p>Given an unsorted array containing n numbers taken from the range 1 to n. The array has some numbers appearing twice, find all these duplicate numbers without using any extra space.</p> <p>Input: nums = [4,3,2,7,8,2,3,1]</p> <p>Output: [2,3]</p> <p>Input: nums = [1,1,2]</p> <p>Output: [1]</p> <p>[3, 4, 4, 5, 5] □ [4, 5]</p> <p>[5, 4, 7, 2, 3, 5, 3] □ [3, 5]</p>
6	Find the Smallest Missing Positive Number	<p>Given an unsorted array containing numbers, find the smallest missing positive number in it. Input: nums = [1,2,0]</p> <p>Output: 3</p> <p>Explanation: The numbers in the range [1,2] are all in the array.</p> <p>Input: nums = [3,4,-1,1]</p> <p>Output: 2</p> <p>Explanation: 1 is in the array but 2 is missing.</p> <p>Input: nums = [7,8,9,11,12]</p> <p>Output: 1</p> <p>Explanation: The smallest positive integer 1 is missing.</p> <p>[-3, 1, 5, 4, 2] □ 3</p> <p>[3, -2, 0, 1, 2] □ 4</p> <p>[3, 2, 5, 1] □ 4</p>

7	Find the First K Missing Positive Numbers	<p>Given an array arr of positive integers sorted in a strictly increasing order, and an integer k.</p> <p>Return the kth positive integer that is missing from this array. Input: arr = [2,3,4,7,11], k = 5</p> <p>Output: 9</p> <p>Explanation: The missing positive integers are [1,5,6,8,9,10,12,13,...]. The 5th missing positive integer is 9.</p> <p>Input: arr = [1,2,3,4], k = 2</p> <p>Output: 6</p> <p>Explanation: The missing positive integers are [5,6,7,...]. The 2nd missing positive integer is 6.</p>
8	Merge Intervals	<p>Given an array of intervals where intervals[i] = [starti, endi], merge all overlapping intervals, and return an array of the non-overlapping intervals that cover all the intervals in the input.</p> <p>Input: intervals = [[1,3],[2,6],[8,10],[15,18]]</p> <p>Output: [[1,6],[8,10],[15,18]]</p> <p>Explanation: Since intervals [1,3] and [2,6] overlap, merge them into [1,6].</p>
		<p>Input: intervals = [[1,4],[4,5]]</p> <p>Output: [[1,5]]</p> <p>Explanation: Intervals [1,4] and [4,5] are considered overlapping.</p> <p>Input: intervals = [[6,7], [2,4], [5,9]]</p> <p>Output: [[2,4], [5,9]],</p> <p>Explanation: Since the intervals [6,7] and [5,9] overlap, we merged them into one [5,9].</p>

9	Insert Interval	<p>You are given an array of non-overlapping intervals intervals where $\text{intervals}[i] = [\text{start}_i, \text{end}_i]$ represent the start and the end of the ith interval and intervals is sorted in ascending order by start_i. You are also given an interval $\text{newInterval} = [\text{start}, \text{end}]$ that represents the start and end of another interval.</p> <p>Insert newInterval into intervals such that intervals is still sorted in ascending order by start_i and intervals still does not have any overlapping</p> <div data-bbox="673 430 1185 714"> <p>time →</p> <p>1) \Rightarrow 'a' and 'b' do not overlap</p> <p>2) \Rightarrow 'a' & 'b' overlap, 'b' ends after 'a'</p> <p>3) \Rightarrow 'a' completely overlaps 'b'</p> <p>4) \Rightarrow 'a' & 'b' overlap, 'a' ends after 'b'</p> <p>5) \Rightarrow 'b' completely overlaps 'a'</p> <p>6) \Rightarrow 'a' and 'b' do not overlap</p> </div> <p>intervals (merge overlapping intervals if necessary).</p> <p>Return intervals after the insertion.</p> <p>Input: intervals = [[1,3],[6,9]], newInterval = [2,5] Output: [[1,5],[6,9]]</p> <p>Input: intervals = [[1,2],[3,5],[6,7],[8,10],[12,16]], newInterval = [4,8] Output: [[1,2],[3,10],[12,16]] Explanation: Because the new interval [4,8] overlaps with [3,5],[6,7],[8,10].</p> <p>Input: intervals = [[1,3], [5,7], [8,12]] newInterval = [4,6] Output: [[1,3], [4,7], [8,12]], Explanation: After insertion, since [4,6] overlaps with [5,7], we merged them into one [4,7].</p> <p>Input: intervals = [[1,3], [5,7], [8,12]] newInterval = [4,10] Output: [[1,3], [4,12]] Explanation: After insertion, since [4,10] overlaps with [5,7] & [8,12], we merged them into [4,12]. Input: intervals = [[2,3],[5,7]] newInterval = [1,4] Output: [[1,4], [5,7]], Explanation: After insertion, since [1,4] overlaps with [2,3], we merged them into one [1,4].</p>
---	-----------------	--

10	Intervals Intersection	<p>You are given two lists of closed intervals, firstList and secondList, where firstList[i] = [starti, endi] and secondList[j] = [startj, endj]. Each list of intervals is pairwise disjoint and in sorted order.</p> <p>Return the intersection of these two interval lists.</p> <p>A closed interval [a, b] (with a <= b) denotes the set of real numbers x with a <= x <= b.</p> <p>The intersection of two closed intervals is a set of real numbers that are either empty or represented as a closed interval. For example, the intersection of [1, 3] and [2, 4] is [2, 3].</p> <p>Example 1:</p>  <p>Input: firstList = [[0,2],[5,10],[13,23],[24,25]], secondList = [[1,5],[8,12],[15,24],[25,26]] Output: [[1,2],[5,5],[8,10],[15,23],[24,24],[25,25]]</p> <p>Example 2:</p> <p>Input: firstList = [[1,3],[5,9]], secondList = [] Output: []</p> <p>Input: firstList = [[1, 3], [5, 6], [7, 9]], secondList = [[2, 3], [5, 7]] Output: [2, 3], [5, 6], [7, 7] The output list contains the common intervals between the two lists.</p> <p>Input: firstList = [[1, 3], [5, 7], [9, 12]], secondList = [[5, 10]] Output: [5, 7], [9, 10] The output list contains the common intervals between the two lists.</p>
----	------------------------	--

11	Conflicting Appointments	<p>Given an array of intervals representing N appointments, find out if a person can attend all the appointments.</p> <p>Input: [[1,4], [2,5], [7,9]] Output: False Since [1,4] and [2,5] overlap, a person cannot attend both of these appointments.</p> <p>Input: [[6,7], [2,4], [8,12]] Output: True None of the appointments overlap, therefore a person can attend all of them.</p> <p>Input: [[4,5], [2,3], [3,6]] Output: False Since [4,5] and [3,6] overlap, a person cannot attend both of these appointments.</p> <p>Input: [[4,5], [2,3], [3,6], [5,7], [7,8]] Output: False, printing conflicts [4,5] and [3,6] conflict. [3,6] and [5,7] conflict.</p>
12	Minimum Meeting Rooms	<p>Given a list of intervals representing the start and endTime of N meetings, find the minimum number of rooms required to hold all the meetings.</p> <p>Input: [[4,5], [2,3], [2,4], [3,5]] Output: 2 Explanation: We will need one room for [2,3] and [3,5], and another room for [2,4] and [4,5].</p> <p>Input: [[1,4], [2,3], [3,6]] Output: 2 Since [1,4] overlaps with the other two meetings [2,3] and [3,6], we need two rooms to hold all the meetings.</p> <p>Input: [[6,7], [2,4], [8,12]] Output: 1 None of the meetings overlap, therefore we only need one room to hold all meetings.</p> <p>Similar Problems: Given a list of intervals, find the point where the maximum number of intervals overlap.</p> <p>Given a list of intervals representing the arrival and departure times of trains to a train station, our goal is to find the minimum number of platforms required for the train station so that no train has to wait.</p>

13	Maximum CPU Load	<p>We are given a list of jobs. Each job has a startTime, an endTime, and a CPU load when it is running. Our goal is to find the Maximum CPU Load at any time if all the jobs are running on the same machine.</p> <p>Jobs: [[1,4,3], [2,5,4], [7,9,6]] Output: 7 Explanation: Since [1,4,3] and [2,5,4] overlap, their maximum CPU load (3+4=7) will be when both the jobs are running at the same time i.e., during the time interval (2,4).</p> <p>Jobs: [[6,7,10], [2,4,11], [8,12,15]] Output: 15 Explanation: None of the jobs overlap, therefore we will take the maximum load of any job which is 15.</p> <p>Jobs: [[1,4,2], [2,4,1], [3,6,5]] Output: 8 Explanation: Maximum CPU load will be 8 as all jobs overlap during the time interval [3,4].</p>
14	Car Pooling	<p>There is a car with capacity empty seats. The vehicle only drives east (i.e., it cannot turn around and drive west).</p> <p>You are given the integer capacity and an array trip where trips[i] = [numPassengers_i, from_i, to_i] indicates that the ith trip has numPassengers_i passengers and the locations to pick them up and drop them off are from_i and to_i respectively. The locations are given as the number of kilometres due east from the car's initial location.</p> <p>Return true if it is possible to pick up and drop off all passengers for all the given trips, or false otherwise.</p> <p>Input: trips = [[2,1,5],[3,3,7]], capacity = 4 Output: false</p> <p>Input: trips = [[2,1,5],[3,3,7]], capacity = 5 Output: true</p>
15	Student Free Time	<p>For K students, we are given a list of intervals representing the studying hours of each student. Our goal is to find out if there is a free interval that is common to all students. You can assume that each list of students studying hours is sorted on the startTime.</p> <p>Input: Student Studying Hours=[[1,3], [9,12]], [[2,4], [6,8]] Output: [4,6], [8,9] Explanation: All students are free between [4,6] and [8,9].</p> <p>Input: Student Studying Hours= [[1,3], [5,6]], [[2,3], [6,8]] Output: [3,5] Explanation: Both students are free between [3,5]</p> <p>Input: Student Studying Hours=[[1,3], [2,4]], [[3,5], [7,9]] Output: [5,7] Explanation: All students are free between [5,7]</p>

Note:

- If Code similarity is found, assignment will not be considered and Zero (0) Marks will be awarded.
- You have to upload a single document consisting of all the above programs and corresponding Output.

Reference:

- <https://emre.me/coding-patterns/cyclic-sort/3>
- <https://emre.me/coding-patterns/merge-intervals/>

1) Find All Duplicates in an Array

Code:

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
public class Assignments {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("");
        String[] input = scanner.nextLine().split(" ");
        int[] nums = new int[input.length];
        for (int i = 0; i < input.length; i++) {
            nums[i] = Integer.parseInt(input[i]);
        }
        List<Integer> result = findDuplicates(nums);
        System.out.println("");
        System.out.println(result);
    }
    public static List<Integer> findDuplicates(int[] nums) {
        List<Integer> result = new ArrayList<>();
        for (int i = 0; i < nums.length; i++) {
            int idx = Math.abs(nums[i]) - 1;
            if (nums[idx] < 0) {
                result.add(idx + 1);
            } else {
                nums[idx] = -nums[idx];
            }
        }
        return result;
    }
}
```

Output:

```
C:\Users\Windows\OneDrive\Documents\CP>javac Assignments.java

C:\Users\Windows\OneDrive\Documents\CP>java Assignments.java
4 3 2 7 8 2 3 1

[2, 3]

C:\Users\Windows\OneDrive\Documents\CP>|
```

2) Find All Numbers Disappeared in an Array Or Find all Missing Numbers

Code:

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
public class Assignments {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("");
        String[] input = scanner.nextLine().split(" ");
        int[] nums = new int[input.length];
        for (int i = 0; i < input.length; i++) {
            nums[i] = Integer.parseInt(input[i]);
        }
        int n = nums.length;
        List<Integer> result = findMissingNumbers(nums, n);
        System.out.println("The missing numbers in the range [1, " + n + "]
are:");
        System.out.println(result);
    }
    public static List<Integer> findMissingNumbers(int[] nums, int n) {
        List<Integer> result = new ArrayList<>();
        for (int i = 1; i <= n; i++) {
            boolean found = false;
            for (int j = 0; j < nums.length; j++) {
                if (nums[j] == i) {
                    found = true;
                    break;
                }
            }
            if (!found) {
                result.add(i);
            }
        }
        return result;
    }
}
```

Output:

```
C:\Users\Windows\OneDrive\Documents\CP>javac Assignments.java

C:\Users\Windows\OneDrive\Documents\CP>java Assignments.java
4 3 2 7 8 2 3 1
The missing numbers in the range [1, 8] are:
[5, 6]

C:\Users\Windows\OneDrive\Documents\CP>
```

3) Set Mismatch Or Find the Corrupt Pair

Code:

```
import java.util.Scanner;

public class Assignments {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("");
        String[] input = scanner.nextLine().split(" ");
        int[] nums = new int[input.length];
        for (int i = 0; i < input.length; i++) {
            nums[i] = Integer.parseInt(input[i]);
        }
        int[] result = findErrorNums(nums);
        System.out.println("");
        System.out.println(result[0] + ", " + result[1]);
    }

    public static int[] findErrorNums(int[] nums) {
        int[] result = new int[2];
        int n = nums.length;
        for (int i = 0; i < n; i++) {
            int index = Math.abs(nums[i]) - 1;
            if (nums[index] < 0) {
                result[0] = index + 1;
            } else {
                nums[index] = -nums[index];
            }
        }
        for (int i = 0; i < n; i++) {
            if (nums[i] > 0) {
```

```

        result[1] = i + 1;
    }
}
return result;
}
}

```

Output:

```

C:\Users\Windows\OneDrive\Documents\CP>javac Assignments.java

C:\Users\Windows\OneDrive\Documents\CP>java Assignments.java
1 2 2 4

2, 3

C:\Users\Windows\OneDrive\Documents\CP>|

```

4) Find the Duplicate Number

Code:

```

import java.util.Scanner;

public class Assignments {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("");
        String[] input = scanner.nextLine().split(" ");
        int[] nums = new int[input.length];
        for (int i = 0; i < input.length; i++) {
            nums[i] = Integer.parseInt(input[i]);
        }
        int duplicate = findDuplicate(nums);
        System.out.println("" + duplicate);
    }

    public static int findDuplicate(int[] nums) {
        int n = nums.length - 1;
        for (int i = 0; i <= n; i++) {
            int index = Math.abs(nums[i]) - 1;
            if (nums[index] > 0) {
                nums[index] = -nums[index];
            } else {

```

```

        return index + 1;
    }
}
return -1;
}
}

```

Output:

```

C:\Users\Windows\OneDrive\Documents\CP>javac Assignments.java

C:\Users\Windows\OneDrive\Documents\CP>java Assignments.java
1 4 4 3 2
4

C:\Users\Windows\OneDrive\Documents\CP>

```

5) Find all Duplicate Numbers

Code:

```

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Scanner;
public class Assignments {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("");
        String[] tokens = input.nextLine().split(" ");
        int[] nums = new int[tokens.length];
        for (int i = 0; i < nums.length; i++) {
            nums[i] = Integer.parseInt(tokens[i]);
        }
        List<Integer> duplicates = findDuplicates(nums);
        System.out.println("" + duplicates);
    }
    public static List<Integer> findDuplicates(int[] nums) {
        List<Integer> duplicates = new ArrayList<>();
        for (int i = 0; i < nums.length; i++) {
            int index = Math.abs(nums[i]) - 1;
            if (nums[index] > 0) {

```

```

        nums[index] = -nums[index];
    } else {
        duplicates.add(Math.abs(nums[i]));
    }
}
return duplicates;
}
}

```

Output:

```

C:\Users\Windows\OneDrive\Documents\CP>javac Assignments.java

C:\Users\Windows\OneDrive\Documents\CP>java Assignments.java

4 3 2 7 8 2 3 1
[2, 3]

C:\Users\Windows\OneDrive\Documents\CP>|

```

6) Find the Smallest Missing Positive Number

Code:

```

import java.util.*;

public class Assignments {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("");
        String[] strArr = input.nextLine().split(" ");
        int[] nums = new int[strArr.length];
        System.out.println("");
        for (int i = 0; i < strArr.length; i++) {
            nums[i] = Integer.parseInt(strArr[i]);
        }
        int result = findSmallestMissingPositive(nums);
        System.out.println(result);
    }
    public static int findSmallestMissingPositive(int[] nums) {
        int n = nums.length;
        boolean containsOne = false;
        for (int num : nums) {

```

```

        if (num == 1) {
            containsOne = true;
            break;
        }
    }
    if (!containsOne) {
        return 1;
    }
    for (int i = 0; i < n; i++) {
        if (nums[i] <= 0 || nums[i] > n) {
            nums[i] = 1;
        }
    }

    for (int i = 0; i < n; i++) {
        int index = Math.abs(nums[i]) - 1;
        if (nums[index] > 0) {
            nums[index] = -nums[index];
        }
    }
    for (int i = 0; i < n; i++) {
        if (nums[i] > 0) {
            return i + 1;
        }
    }
    return n + 1;
}
}

```

Output:

```

C:\Users\Windows\OneDrive\Documents\CP>javac Assignments.java
C:\Users\Windows\OneDrive\Documents\CP>java Assignments.java
1 2 0
3
C:\Users\Windows\OneDrive\Documents\CP>|

```

7) Find the First K Missing Positive Numbers

Code:

```
import java.util.Scanner;
public class Assignments {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("");
        int n = sc.nextInt();
        int[] arr = new int[n];
        System.out.println("");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }
        System.out.print("");
        int k = sc.nextInt();
        int missing = 1, i = 0;
        while (k > 0 && i < n) {
            if (arr[i] == missing) {
                missing++;
            } else {
                k -= arr[i] - missing + 1;
                missing = arr[i] + 1;
            }
            i++;
        }
        if (k > 0) {
            missing += k - 1;
        }
        System.out.println("The " + k + "th missing positive integer is " +
missing);
    }
}
```

Output:

```
C:\Users\Windows\OneDrive\Documents\CP>javac Assignments.java
C:\Users\Windows\OneDrive\Documents\CP>java Assignments.java
4
1 2 3 4
2
The 2th missing positive integer is 6
C:\Users\Windows\OneDrive\Documents\CP>
```


8) Merge Intervals

Code:

```
import java.util.*;

public class Assignments {
    public int[][] merge(int[][] intervals) {
        Arrays.sort(intervals, (a, b) -> Integer.compare(a[0], b[0]));
        List<int[]> merged = new ArrayList<>();
        for (int[] interval : intervals) {
            if (merged.isEmpty() || merged.get(merged.size()-1)[1] < interval[0])
            {
                merged.add(interval);
            } else {
                merged.get(merged.size()-1)[1] =
                Math.max(merged.get(merged.size()-1)[1], interval[1]);
            }
        }
        return merged.toArray(new int[merged.size()][]);
    }

    public static void main(String[] args) {
        int[][] intervals = {{1,3},{2,6},{8,10},{15,18}};
        Assignments mi = new Assignments();
        int[][] mergedIntervals = mi.merge(intervals);
        System.out.println(Arrays.deepToString(mergedIntervals));
    }
}
```

Output:

```
C:\Users\Windows\OneDrive\Documents\CP>javac Assignments.java

C:\Users\Windows\OneDrive\Documents\CP>java Assignments.java
[[1, 6], [8, 10], [15, 18]]

C:\Users\Windows\OneDrive\Documents\CP>|
```

9) Insert Interval

Code:

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
public class Assignments {
    public int[][] insert(int[][] intervals, int[] newInterval) {
        List<int[]> merged = new ArrayList<>();
        int i = 0, n = intervals.length;
        while (i < n && intervals[i][1] < newInterval[0]) {
            merged.add(intervals[i++]);
        }
        while (i < n && intervals[i][0] <= newInterval[1]) {
            newInterval[0] = Math.min(newInterval[0], intervals[i][0]);
            newInterval[1] = Math.max(newInterval[1], intervals[i][1]);
            i++;
        }
        merged.add(newInterval);
        while (i < n) {
            merged.add(intervals[i++]);
        }
        int[][] res = new int[merged.size()][2];
        for (i = 0; i < merged.size(); i++) {
            res[i] = merged.get(i);
        }
        return res;
    }
    public static void main(String[] args) {
        Assignments solution = new Assignments();
        int[][] intervals1 = {{1,3},{6,9}};
        int[] newInterval1 = {2,5};
        int[][] res1 = solution.insert(intervals1, newInterval1);
        System.out.println(Arrays.deepToString(res1));
        int[][] intervals2 = {{1,2},{3,5},{6,7},{8,10},{12,16}};
        int[] newInterval2 = {4,8};
        int[][] res2 = solution.insert(intervals2, newInterval2);
        System.out.println(Arrays.deepToString(res2));
        int[][] intervals3 = {{1,3},{5,7},{8,12}};
        int[] newInterval3 = {4,6};
    }
}
```

```

        int[][] res3 = solution.insert(intervals3, newInterval3);
        System.out.println(Arrays.deepToString(res3));
    }
}

```

Output:

```

C:\Users\Windows\OneDrive\Documents\CP>javac Assignments.java

C:\Users\Windows\OneDrive\Documents\CP>java Assignments.java
[[1, 5], [6, 9]]
[[1, 2], [3, 10], [12, 16]]
[[1, 3], [4, 7], [8, 12]]

C:\Users\Windows\OneDrive\Documents\CP>|

```

10) Intervals Intersection

Code:

```

import java.util.*;

public class Assignments {

    public static void main(String[] args) {
        int[][] firstList = {{0, 2}, {5, 10}, {13, 23}, {24, 25}};
        int[][] secondList = {{1, 5}, {8, 12}, {15, 24}, {25, 26}};

        int[][] result = intervalIntersection(firstList, secondList);

        System.out.println("");
        for (int[] interval : result) {
            System.out.print(Arrays.toString(interval) + " ");
        }
    }

    public static int[][] intervalIntersection(int[][] firstList, int[][] secondList) {
        int i = 0;
        int j = 0;
        List<int[]> result = new ArrayList<>();

        while (i < firstList.length && j < secondList.length) {
            int start = Math.max(firstList[i][0], secondList[j][0]);
            int end = Math.min(firstList[i][1], secondList[j][1]);

```

```

        if (start <= end) {
            result.add(new int[] {start, end});
        }

        if (firstList[i][1] < secondList[j][1]) {
            i++;
        } else {
            j++;
        }
    }

    return result.toArray(new int[result.size()][2]);
}
}

```

Output:

```

C:\Users\Windows\OneDrive\Documents\CP>javac Assignments.java

C:\Users\Windows\OneDrive\Documents\CP>java Assignments.java

[1, 2] [5, 5] [8, 10] [15, 23] [24, 24] [25, 25]
C:\Users\Windows\OneDrive\Documents\CP>

```

11) Conflicting Appointments:

Code:

```

import java.util.Arrays;

public class Assignments {
    public static boolean canAttendAllAppointments(int[][] intervals) {
        if (intervals == null || intervals.length == 0) {
            return false;
        }
        Arrays.sort(intervals, (a, b) -> Integer.compare(a[0], b[0]));

        for (int i = 1; i < intervals.length; i++) {
            if (intervals[i][0] <= intervals[i - 1][1]) {
                return false;
            }
        }
    }
}

```

```

        return true;
    }
    public static void main(String[] args) {
        int[][] intervals1 = {{1, 4}, {2, 5}, {7, 9}};
        int[][] intervals2 = {{6, 7}, {2, 4}, {8, 12}};
        int[][] intervals3 = {{4, 5}, {2, 3}, {3, 6}};
        System.out.println(canAttendAllAppointments(intervals1)); // false
        System.out.println(canAttendAllAppointments(intervals2)); // true
        System.out.println(canAttendAllAppointments(intervals3)); // false
    }
}

```

Output:

```

C:\Users\Windows\OneDrive\Documents\CP>javac Assignments.java

C:\Users\Windows\OneDrive\Documents\CP>java Assignments.java
false
true
false

C:\Users\Windows\OneDrive\Documents\CP>|

```

12) Minimum Meeting Rooms:

Code:

```

import java.util.Arrays;
import java.util.PriorityQueue;
public class Assignments {
    public static int minMeetingRooms(int[][] intervals) {
        Arrays.sort(intervals, (a, b) -> a[0] - b[0]);
        PriorityQueue<Integer> queue = new PriorityQueue<>();
        for (int[] interval : intervals) {
            if (!queue.isEmpty() && interval[0] >= queue.peek()) {
                queue.poll();
            }
            queue.offer(interval[1]);
        }
        return queue.size();
    }
}

```

```

public static void main(String[] args) {
    int[][] intervals1 = {{4,5}, {2,3}, {2,4}, {3,5}};
    int[][] intervals2 = {{1,4}, {2,3}, {3,6}};
    int[][] intervals3 = {{6,7}, {2,4}, {8,12}};
    System.out.println(minMeetingRooms(intervals1)); // Output: 2
    System.out.println(minMeetingRooms(intervals2)); // Output: 2
    System.out.println(minMeetingRooms(intervals3)); // Output: 1
}
}

```

Output:

```

C:\Users\Windows\OneDrive\Documents\CP>javac Assignments.java

C:\Users\Windows\OneDrive\Documents\CP>java Assignments.java
2
2
1

C:\Users\Windows\OneDrive\Documents\CP>|

```

13) Maximum CPU Load:

Code:

```

import java.util.*;
public class Assignments {
    static class Job {
        int start;
        int end;
        int load;
        public Job(int start, int end, int load) {
            this.start = start;
            this.end = end;
            this.load = load;
        }
    }
    public static int findMaxCPULoad(List<Job> jobs) {
        Collections.sort(jobs, (a, b) -> Integer.compare(a.start, b.start));
        int maxCPULoad = 0;
        int currentCPULoad = 0;
        PriorityQueue<Job> minHeap = new PriorityQueue<>(jobs.size(), (a,
b) -> Integer.compare(a.end, b.end));
    }
}

```

```

    for (Job job : jobs) {
        while (!minHeap.isEmpty() && job.start >= minHeap.peek().end) {
            currentCPULoad -= minHeap.poll().load;
        }
        minHeap.offer(job);
        currentCPULoad += job.load;
        maxCPULoad = Math.max(maxCPULoad, currentCPULoad);
    }
    return maxCPULoad;
}

public static void main(String[] args) {
    List<Job> jobs1 = Arrays.asList(new Job(1, 4, 3), new Job(2, 5, 4), new
Job(7, 9, 6));
    List<Job> jobs2 = Arrays.asList(new Job(6, 7, 10), new Job(2, 4, 11),
new Job(8, 12, 15));
    List<Job> jobs3 = Arrays.asList(new Job(1, 4, 2), new Job(2, 4, 1), new
Job(3, 6, 5));
    System.out.println(findMaxCPULoad(jobs1));
    System.out.println(findMaxCPULoad(jobs2));
    System.out.println(findMaxCPULoad(jobs3));
}
}

```

Output:

```

C:\Users\Windows\OneDrive\Documents\CP>javac Assignments.java

C:\Users\Windows\OneDrive\Documents\CP>java Assignments.java
7
15
8

C:\Users\Windows\OneDrive\Documents\CP>|

```

14) Car Pooling:

Code:

```
import java.util.Arrays;
public class Assignments {
    public boolean canCarryAllPassengers(int[][] trips, int capacity) {
        int n = trips.length;
        int[][] events = new int[2 * n][2];
        for (int i = 0; i < n; i++) {
            events[2 * i] = new int[] {trips[i][1], trips[i][0]};
            events[2 * i + 1] = new int[] {trips[i][2], -trips[i][0]};
        }
        Arrays.sort(events, (a, b) -> a[0] - b[0]);
        int currentPassengers = 0;
        for (int[] event : events) {
            currentPassengers += event[1];
            if (currentPassengers > capacity) {
                return false;
            }
        }
        return true;
    }

    public static void main(String[] args) {
        Assignments carTrips = new Assignments();
        int[][] trips1 = {{2,1,5},{3,3,7}};
        int capacity1 = 4;
        System.out.println(carTrips.canCarryAllPassengers(trips1, capacity1));
        int[][] trips2 = {{2,1,5},{3,3,7}};
        int capacity2 = 5;
        System.out.println(carTrips.canCarryAllPassengers(trips2, capacity2));
    }
}
```

Output:

```
C:\Users\Windows\OneDrive\Documents\CP>javac Assignments.java
C:\Users\Windows\OneDrive\Documents\CP>java Assignments.java
false
true
C:\Users\Windows\OneDrive\Documents\CP>
```


15) Student Free Time:

Code:

```
import java.util.*;
public class Assignments {
    public static List<Interval> findFreeTime(List<List<Interval>>
schedules) {
        List<Interval> result = new ArrayList<>();
        if (schedules == null || schedules.size() == 0)
            return result;
        List<Interval> mergedIntervals = new ArrayList<>();
        for (List<Interval> schedule : schedules) {
            mergedIntervals = mergeIntervals(mergedIntervals, schedule);
        }
        for (int i = 1; i < mergedIntervals.size(); i++) {
            Interval current = mergedIntervals.get(i);
            Interval previous = mergedIntervals.get(i - 1);
            if (current.start > previous.end) {
                result.add(new Interval(previous.end, current.start));
            }
        }
        return result;
    }
    private static List<Interval> mergeIntervals(List<Interval> intervals1,
List<Interval> intervals2) {
        List<Interval> mergedIntervals = new ArrayList<>();
        int i = 0, j = 0;
        while (i < intervals1.size() && j < intervals2.size()) {
            Interval interval1 = intervals1.get(i);
            Interval interval2 = intervals2.get(j);
            if (interval1.start <= interval2.start) {
                mergedIntervals.add(interval1);
                i++;
            } else {
                mergedIntervals.add(interval2);
                j++;
            }
        }
        while (i < intervals1.size()) {
            mergedIntervals.add(intervals1.get(i));
        }
    }
}
```

```

        i++;
    }
    while (j < intervals2.size()) {
        mergedIntervals.add(intervals2.get(j));
        j++;
    }
    return mergedIntervals;
}

public static void main(String[] args) {
    List<List<Interval>> schedules = new ArrayList<>();
    schedules.add(Arrays.asList(new Interval(1, 3), new Interval(9, 12)));
    schedules.add(Collections.singletonList(new Interval(2, 4)));
    schedules.add(Collections.singletonList(new Interval(6, 8)));
    System.out.println(findFreeTime(schedules));
}
}

class Interval {
    int start;
    int end;
    Interval(int start, int end) {
        this.start = start;
        this.end = end;
    }
    @Override
    public String toString() {
        return "[" + start + ", " + end + "]";
    }
}

```

Output:

```

C:\Users\Windows\OneDrive\Documents\CP>javac Assignments.java

C:\Users\Windows\OneDrive\Documents\CP>java Assignments.java
[[4, 6], [8, 9]]

C:\Users\Windows\OneDrive\Documents\CP>|

```