

```

public class RangeCombiner {
    private List<Double> mins = new ArrayList<>();
    private List<Double> maxs = new ArrayList<>();

    public void addRange(double min, double max) {
        if (max < min) { return; }
        for (int i = 0; i < mins.size(); i++) {
            if (max <= mins.get(i) || min <= maxs.get(i)) {
                insertValueAtIndexAndFixForward(min, max, i);
                return;
            }
        }
        // Did not end up merging, new range goes to end
        mins.add(min); maxs.add(max);
    }

    private void insertValueAtIndexAndFixForward(double currMin, double currMax, int i) {
        mins.add(i, currMin); maxs.add(i, currMax);
        // Need to possibly merge it with followup ranges
        // As long as i is not the last index:
        while (i + 1 < mins.size()) {
            Double nextMin = mins.get(i + 1);
            Double nextMax = maxs.get(i + 1);
            if (rangesOverlap(currMin, currMax, nextMin, nextMax)) {
                currMin = Math.min(currMin, nextMin);
                currMax = Math.max(currMax, nextMax);
                mins.set(i, currMin); maxs.set(i, currMax);
                mins.remove(i + 1); maxs.remove(i + 1);
            } else { break; }
        }
    }

    private boolean rangesOverlap(double min1, double max1, double min2, double max2) {
        return (max1 >= min2 && min1 <= min2) || (max2 >= min1 && min2 <= min1);
    }

    boolean isRangeOrderValid() {
        for (int i = 0; i < mins.size() - 1; i++) {
            if (maxs.get(i) >= mins.get(i + 1)) { return false; }
        }
        return true;
    }

    private void printRanges() {
        for (int i = 0; i < mins.size(); i++) {
            System.out.println(String.format("%.2f---%.2f", mins.get(i), maxs.get(i)));
        }
    }

    public static void main(String[] args) {
        RangeCombiner combiner = new RangeCombiner();
        combiner.addRange(2.4, 3.7);
        combiner.addRange(5.6, 5.7);
        combiner.addRange(3.5, 3.8);
        combiner.addRange(6.3, 5.7); // empty range, should ignore
        combiner.addRange(5.7, 5.9);
        combiner.addRange(3.9, 4.1);
        combiner.addRange(3.7, 3.9);
        combiner.addRange(1.1, 1.4); // should appear first
    }
}

```

```

        if (!combiner.isRangeOrderValid()) { System.out.println("Invalid_order!"); }
        combiner.printRanges(); // Should print 1.10--1.40, 2.40--4.10 and 5.60--5.90
    }
}

// Transformed
public class RangeCombiner {
    private List<Range> ranges = new ArrayList<>();

    public void addRange(double min, double max) {
        addRangeInternal(new Range(min, max));
    }

    private void addRangeInternal(Range range) {
        if (range.isEmpty()) { return; }
        for (int i = 0; i < ranges.size(); i++) {
            if (range.precedes(ranges.get(i)) || range.doesNotFollow(ranges.get(i))) {
                ranges.add(i, range);
                fixForwardFromIndex(i);
                return;
            }
        }
        ranges.add(range);
    }

    private void fixForwardFromIndex(int i) {
        while (i + 1 < ranges.size() && ranges.get(i).overlapsWith(ranges.get(i + 1))) {
            ranges.set(i, ranges.get(i).mergedWith(ranges.get(i + 1)));
            ranges.remove(i + 1);
        }
    }

    boolean isRangeOrderValid() {
        for (int i = 0; i < ranges.size() - 1; i++) {
            if (ranges.get(i + 1).doesNotFollow(ranges.get(i))) { return false; }
        }
        return true;
    }

    private void printRanges() {
        for (Range range : ranges) { System.out.println(range.format()); }
    }

    public static void main(String[] args) {
        RangeCombiner combiner = new RangeCombiner();
        combiner.addRange(2.4, 3.7);
        combiner.addRange(5.6, 5.7);
        combiner.addRange(3.5, 3.8);
        combiner.addRange(6.3, 5.7); // empty range, should ignore
        combiner.addRange(5.7, 5.9);
        combiner.addRange(3.9, 4.1);
        combiner.addRange(3.7, 3.9);
        combiner.addRange(1.1, 1.4); // should appear first
        if (!combiner.isRangeOrderValid()) { System.out.println("Invalid_order!"); }
        combiner.printRanges(); // Should print 1.10--1.40, 2.40--4.10 and 5.60--5.90
    }

    private static class Range {
        private final double min;
        private final double max;

        private Range(double min, double max) {
            this.min = min;

```

```

    this.max = max;
}

private boolean overlapsWith(Range range) {
    return this.doesNotFollow(range) && range.doesNotFollow(this);
}
private Range mergedWith(Range range) {
    return new Range(Math.min(min, range.min), Math.max(max, range.max));
}
private boolean isEmpty() { return max < min; }
private boolean precedes(Range range) { return max <= range.min; }
private boolean doesNotFollow(Range range) { return min <= range.max; }
private String format() { return String.format("%.2f—%.2f", min, max); }
}
}

```