**ZipCode Merge Test**

This is a cradle project to find and limit the number of zip code ranges provided as input which can be used for restricting deliveries.

**Problem Statement:**

BACKGROUND Sometimes items cannot be shipped to certain zip codes, and the rules for these restrictions are stored as a series of ranges of 5 digit codes. For example if the ranges are:

([49679, 52015], [49800, 50000], [51500, 53479], [45012, 46937], [54012, 59607], [45500, 45590], [45999, 47900], [44000, 45000], [43012, 45950])

Then the item can be shipped to zip code 50200, 51001, and 65532, but cannot be shipped to 51333, 54650, 47230, 47900, or 59600.

Any item might be restricted based on multiple sets of these ranges obtained from multiple sources.

PROBLEM Given a collection of 5-digit ZIP code ranges (each range includes both their upper and lower bounds), provide an algorithm that produces the minimum number of ranges required to represent the same restrictions as the input.

NOTES

* The ranges above are just examples, your implementation should work for any set of arbitrary ranges
* Ranges may be provided in arbitrary order
* Ranges may or may not overlap
* Your solution will be evaluated on the correctness and the approach taken, and adherence to coding standards and best practices

EXAMPLES: If the input = ([49679, 52015], [49800, 50000], [51500, 53479], [45012, 46937], [54012, 59607], [45500, 45590], [45999, 47900], [44000, 45000], [43012, 45950])

Then the output should be = [43012, 47900], [49679, 53479], [54012, 59607]

If the input = [94133,94133] [94200,94299] [94226,94399] Then the output should be = [94133,94133] [94200,94399]

Evaluation Guidelines: Your work will be evaluated against the following criteria:

* Successful implementation
* Efficiency of the implementation
* Design choices and overall code organization
* Code quality and best practices

**Assumptions made:**

1. Assumed input to come as String For instance input [94133,94133] [94200,94299] [94600,94699] as whole under a string.

**DataStructures used:**

1. LinkedList – The approach was to use 2 linkedlist where one was for storing the input and the other was for storing the output after merging the sorted result. Future work will be to use only one LinkedList for doing merge operation.
2. Created maven project to help entension of this project for future work

**Java File Description:**

1. Application.java --> Main driver than reads the input and drives the zipCode processor
2. ZipCode.java --> POJO class to store the lower bound and upper bound of zipcode
3. ZipCodeComparator.java --> To sort based on the lower bound of zipcode from the list
4. ZipCodeProcessor.java --> Loads the final sorted list of zipcodes to the output linked list after validating the input
5. ZipCodeCombiner.java --> Main logic that merges the zipcode ranges and returns the final list

**Tests:**

1. Used Random Integer generator to generate data sets
2. Wrote Junit tests to validate different scenarios of input