**Q1 Analyzing a Graph with Hadoop/Java**

***(a):***

1. Parse each line into a string list. Go through each string list, set the first element as key, third element as value. After the mapper phase, we output key-value pair as follows:

<100,7>

<200,7>

<110,2>

<100,30>

<110,67>

<10,15>

2. Hadoop sorts and aggregates by key, the reducer input is as follows:

<10,15>

<100, [7,30]>

<110, [2,67]>

<200,7>

3. Then for each key-values pair, we loop through the values to find the maximum value max. Then save <key-max> as the output. The output:

10 15

100 30

110 67

200 7

***(b):***

1. Parse each line into a string list. Define a new class GraphPair, pass the string list values to a GraphPair class as composite key, and set null for the value:

<[100 10 7], null>

<[200 10 7], null>

<[110 10 2], null>

<[100 130 30], null>

<[110 130 67], null>

<[10 101 15], null>

2. Overwrite the compareTo() method in GraphPair class. First by receiver ID (ascending), second by weight (descending), and lastly sender ID(ascending). Then define the Partitioner class to make sure that composite keys with the same receiver ID get to the same reducer. The input for reducer(here suppose there's only one reducer):

<[100 10 7], null>

<[200 10 7], null>

<[110 10 2],null>

<[10 101 15], null>

<[110 130 67], null>

<[100 130 30], null>

3. Define the GroupPartitioner class for reducer. Group GraphPair with the same receiver node ID together. Output:

```
------------------------
<[100 10 7], null>
<[200 10 7], null>
<[110 10 2],null>
-----------------------
<[10 101 15], null>
------------------------
<[110 130 67], null>
<[100 130 30], null>
-------------------------
```

4. Output the receiver node ID and the first sender node ID for each group.

<10, 100>

<101, 10>

<130, 110>