Task 1 Part A

MAPPING FUNCTION:

The mapping function for this code takes in the 3 column (src, tgt, weight) from the .tsv File and sets them to itr.nextToken. This means that the StringTokenizer which iterates through all of the tokens will recognize these 3 different data points as tokens and iterate through them. The mapper function takes in src as a Text, tgt as a Text, and weight as an IntWritable. Context.write(tgt, weight) maps the original dataset to a different structure consisting of only a Text column called tgt and an Intwritable column called weight.

REDUCE FUNCTION:

Then, the reduce function simply iterates through the values of the particular key, and sets max = highest value for that key by changing max if the value of value is greater than max. After a full iteration, max will be equal to the highest value for the particular key. For the inline example,

Key: 51 Values: 1, 1, 3 :

Max = 1

1 !> 1 so max = 1

3 > 1 so max = 3

Therefore, max value for 51 is 3, so context.write(key, max) where the key is 51 and max is 3.
This process is repeated for all keys (i.e. 151 and 130), and context.write() will create a full list of max values for each key-value:

OUTPUT:

51	3
151	79
130	10

Task 1 Part B

The algorithm for this problem is as follows (written in pseudo code):

```
MapReduce([]src, []tgt) {
  For (I in []src) {
    Int s = src[i]
    Int t = tgt[i]
    For (j in []tgt) {
      If(tgt[j] = s and tgt[j] != t) {
        Int s2 = src[j]
        Context.write(s2, t)
    }}}}
```

- We have an array of targets and an array of sources, which are the same length- this is key
- We iterate through every element j in the target array
- Then, if the target[j] = src[j], we have found a 2-edge connection
- Also, tgt[j] cannot be equal to itself because of the constraints of the problem
- Then, we output src[j] and tgt[i], where the tgt[j] and src[i] are equal which means they represent the middle node