Code Description

1. Run and test environment:

- Local test environment: JDK8 and Hadoop-2.7.7 (windows10)
- Cluster environment: JDK8 and Hadoop-2.7.7 (centos, 1 master & 2 slaves)

2. Task 1 Common Word

The work I did is in `Mapper1`, `Mapper2`, `CountCommonReducer`, `SortMapper`, `SortReducer` and `ReverseComparator`.

- `Mapper1` and `Mapper2` are to add label on the frequency to tell which passage the frequency belongs to. Please note that only the key codes are shown in the report.

```
//Mapper1
context.write(key,new Text(value.toString()+"_s1"));   //add “_s1” for passage 1
//Mapper2
context.write(key,new Text(value.toString()+"_s2"));   //add “_s2” for passage 2
```

- `CountCommonReducer` is to compare the frequencies of a word (key) for the 2 passages and keep the smaller one. The output is <word, smaller frequency>.

```
//CountCommonReducer
Integer count1 = Integer.MAX_VALUE;
        Integer count2 = 0;
        //keep the smallest frequency in count1
        for(Text v : values){
            String str = v.toString();
            count1 = (Integer.valueOf(str.substring(0,str.length()-3))<count1)?Integer.valueOf(str.substring(0,str.length()-3)):count1;
            count2 += 1;
        }

        if (count2>1){
            context.write(key, new IntWritable(count1));
        }
```

- `SortMapper` and `SortReducer` is to switch the key and value and use the sort of Mappreduce to sort the result automatically. Note that the default sort is in ascending order. We rewrite the `WritableComparator` in hadoop.io as `ReverseComparator`. We just reverse the return value of the comparator to make a descending order.

```
//SortMapper
context.write(new IntWritable(Integer.valueOf(value.toString())),key);      //switch key and value
//SortReducer
for(Text v : values){
                context.write(key,v);
            }
//ReverseComparator
@Override
        public int compare(byte[] b1, int s1, int l1, byte[] b2, int s2, int l2) {
            return (-1)* TEXT_COMPARATOR.compare(b1, s1, l1, b2, s2, l2);    //reverse the return result
        }

        @SuppressWarnings("rawtypes")
        @Override
        public int compare(WritableComparable a, WritableComparable b) {
            if (a instanceof Text && b instanceof Text) {
                return (-1)*(((Text) a).compareTo((Text) b));               //reverse the return result
            }
            return super.compare(a, b);
        }
```