Create a local parallel MapReduce program, counting the total number of each word in different files (ascii format)

- Create N number of mapper process (N should be less than the number of files, so you don't have to split file), create M number of reducer process, the execution can be in the following way (pseudo code):

```
// mapper execution
for i in range(N):
    // you don't need to use map as function name
    thread(map(input_files))

// wait for all mapper thread finish
wait

// reducer execution
for i in range(M):
    // you don't need to use reduce as function name
    thread(reduce(intermediate_files))
```

- Choose N and M by yourself (it's not super important which value you choose, but at least bigger than 2)
- Run mapper process to separate text into words, one word per line, separate them into different buckets (files), then aggregate (count) them in reducer
- Save the map results to files in format such as: mr-N-X, N is the index of mapper process, and X is the "bucket number" for each word: you can use the first letter of each word % M to calculate X
- Reducer X reads all mr-0-X, mr-1-X, … mr-M-X files, aggregate, then output out-X file, any given words should only appear in only one of the out-n files.
- Watch out to remove the comma and period and so on, counting only alphanumeric
- Final format should be as follows, as an example (in each out-X files):

```
Hello: 213
World: 423
....
```

Look at the following figure for some hints: