# 1. Result:

## result as two sentences:

1. to be is to do
2. to do is to be

```
yunke_zhu@cluster-dade-m:~/quiz3$ cat zen1.txt
to be is to do
yunke_zhu@cluster-dade-m:~/quiz3$ cat zen2.txt
to do is to be
yunke_zhu@cluster-dade-m:~/quiz3$ cat zen1.txt zen2.txt | ./mapper.py |./reducer.py |sort
be|is   1
do|is   1
is|to   2
to|be   2
to|do   2
yunke_zhu@cluster-dade-m:~/quiz3$
```

## result as two sentences:

1.How much wood would a woodchuck chuck if a woodchuck could chuck wood?

2.He would chuck, he would, as much as he could, and chuck as much as a woodchuck would If a woodchuck could chuck wood.

```
yunke_zhu@cluster-dade-m:~/quiz3$ cat wood.txt |./mapper.py |./reducer.py |sort
and|chuck       1
as|a    1
as|he   1
as|much 2
a|woodchuck     4
chuck|as        1
chuck|he        1
chuck|if        1
chuck|wood      2
could|and       1
could|chuck     2
he|could        1
he|would        2
how|much        1
if|a    2
much|as 2
much|wood       1
woodchuck|chuck 1
woodchuck|could 2
woodchuck|would 1
wood|would      1
would|a 1
would|as        1
would|chuck     1
would|if        1
yunke_zhu@cluster-dade-m:~/quiz3$
```

# 2. Code:

### Mapper.py:

```
#!/usr/bin/env python
import sys
import os
import json
import re
import subprocess
class Mapper:


    def MAP(self):
#--- get all lines from stdin ---

        pattern = re.compile("[a-zA-Z][a-zA-Z0-9]*")

        for line in sys.stdin:
            line = line.strip()
            words = line.split()

            for i in range(len(words)-1):
                word1 = re.sub(r'[^\w]','', words[i]).lower()
                word2 = re.sub(r'[^\w]','', words[i+1]).lower()
                print '%s\t%s' % (word1+"|"+word2, "1")









exp = Mapper()
exp.MAP()
```

## Reducer.py

```
#!/usr/bin/env python
import sys
word2count = {}
word2count = {}
#--- get all lines from stdin ---
for line in sys.stdin:
    #--- remove leading and trailing whitespace---
    line = line.strip()
    repeated = 0

    #--- split the line into words ---
    words = line.strip().split('\t')
    count = words[1]
    word = words[0]
#    name = words[1]

    try:
        count = int(count)

    except ValueError:
        continue


    try:
        word2count[word] = word2count[word]+count
    except:
        word2count[word] = count


for word in word2count.keys():
        print '%s\t%s'% ( word, word2count[word] )
```

# 3.Conclusion

I just adjust the mapper function and parse two neighbors linked with "|" as one key and "1" as value to the reducer. And we don't need to change any code in reducer.py for this case

# 4. Command

```
cat filename.txt | ./mapper.py | ./reducer.py | sort
```