Exercise 3

3A

Using the output from EX 2 in the form of `word<tab>counter` pairs
we did the following steps:

step 1 DATA

We loaded the merged output file `data.txt` of the word counter to another folder into the hadoop dfs, allowing us to use the it in another hadoop MapReduce framework operation.

step 2 MAP REDUCE

We needed to use the counter to be key key and the word as value for this particular task so that's what we did in the mapper:

```
#!/usr/bin/env python
"""mapper.py"""

import sys
import re

# input comes from STDIN (standard input)
# in the form of <word><\t><count>
for line in sys.stdin:
    word, count = line.split('\t')
    word = word.strip()
    if len(word) > 0 :
        try:
            count = int(count)

        except ValueError:
            # count was not a number, so silently
            # ignore/discard this line
            print "int cast failed!"
            continue

        # swapping key-value to take advantage of the
        # map reduce framework shuffle reordering by key
        print '%d\t%s' % (count, word)
```

The reducer funtion aggregate the data ordered in the automatic shuffle phase. The ordering is done automatically on the key which in this case is the counter, allowing us to fill a list of words having the same counter.

```
#!/usr/bin/env python
"""reducer.py"""

from operator import itemgetter
import sys

current_words = []
current_count = 0

# input comes from STDIN
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # parse the input we got from mapper.py
    # <count> <word>
    count, word = line.split('\t')

    # convert count (currently a string) to int
    try:
        count = int(count)

    except ValueError:
        # count was not a number, so silently
        # ignore/discard this line
        continue

    # if first value
    if current_count == 0:
        current_count = count

    # this IF-switch only works because Hadoop sorts map output
    # by key (here: word) before it is passed to the reducer
    if current_count == count:
        current_words.append(word)

    else:
        print '%d\t%s'%(current_count,"\t".join(current_words))
        current_count = count
        current_words = [word]

# do not forget to output the last record <--
if current_count == count:
    print '%d\t%s'%(count,"\t".join(current_words))
```

step 3 RUNNING TASK

We ran the task this way

```
hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-*streaming*.jar -mapper mapper.py -file ./mapper.py -r
```

step 4 RESULTS

After merging with `hdfs dfs getmerge` we observed the results:

```
[user_lsc_3@it EX3]$ tail output.txt
94      madame  threw   stuff   seized  glasses bill
95      listen  closed  azkaban firebolt        dragon  cauldron        girl    free    whether seem    ready   w
96      task    hope    ah      dog     dropped
97      heads   christmas       friends days    third   rita    stairs
970     ve
973     when
977     know
98      heart   big     maxime  such    near
983     just
99      fact    entered maybe
```