Exercise 3

3C

Using the output from EX 3A in the form of `counter<tab>[words]` we did the following steps:

step 1 DATA

We loaded the merged output file from 3A namely `data.txt` into the hdfs

step 2 MAP-REDUCE

We used the same mapper of ex 3A, having the keys to be the word frequencies and the values a list of words separated by a tab

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

import sys

# input comes from STDIN (standard input)
# in the form of <word><\t><count>
for line in sys.stdin:
    words = line.split('\t')

    if len(words) > 0 :
        try:
            count = int(words[0])

        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 '%s' % ("\t".join(words))
```

As for 3A, 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.

This time we needed a new variable to store the maximum key and the list of words associated with it.

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

from operator import itemgetter
import sys

max = 0
max_words = []

# 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>
    values  = line.split('\t')

    # convert count (currently a string) to int
    try:
        key = int(values[0])
        if key <= max:
            continue
    except ValueError:
        continue

    if key <= max:
        continue

    else:
        max = key
        max_words = values


print '%s'%("\t".join(max_words))
```

step 3 RUNNIN TASK

we wanted to make sure the algorithm worked before running it on the cluster in the example data first so we simulated it locally:

```
[user_lsc_3@it C]$ cat example.txt
3       pen     car     house   glass
5       battery phone
6       the
89      hi
[user_lsc_3@it C]$ cat example.txt | python mapper.py | sort -k1,1 | python reducer.py
89      hi
```

We ran the task the usual way using the exercise 3A data already loaded

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

step 4

After merging with `hdfs dfs getmerge` we sorted by occurence of the key (which is a word frequency) and observed the obteined occurrence of frequencies in the harry potter books:

```
[user_lsc_3@it C]$ hdfs dfs -getmerge ex3c/output output.txt
2021-12-09 03:39:45,705 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... usi
[user_lsc_3@it C]$ cat output.txt
22046   the
```

the most common word is `the` .