EX 1

`wc1.py` with a map-reduce scheme to select the 3 words that occur more frequently in the dataset:

We wrote the following priogram using map-filter-reduce pattern.
we used `take()` to get the first three elements after ordering by ke in a descending order.

```
# create a program wc1.py with a map-reduce scheme to select the
# 3 words that occur more frequently in the dataset

import sys
import re

from pyspark import SparkContext, SparkConf

_DATA_ = "hdfs:/user/user_lsc_3/labPySparkData/big.txt"
_OUTPUT_ = "hdfs:/user/user_lsc_3/labPySparkData/output"


def isWord(x):
    if re.match(r"[A-Za-z]*",x):
       return True

    return False

if __name__ == "__main__":

    # create Spark context with necessary configuration

    sc = SparkContext("local","PySpark Word Count Exmaple")

    # read data from text file and split each line into words
    rdd = sc.textFile(_DATA_).flatMap(lambda line: re.split(r"[^\w]*", line.strip().lower()))

    wordCounts = rdd.map(lambda word: (word.lower(), 1)).reduceByKey(lambda a,b:a +b)

    top3 = sc.parallelize(wordCounts.map(lambda x: (x[1], x[0])).sortByKey(ascending=False).take

    top3.saveAsTextFile(_OUTPUT_)
```

We use the regex `[^A-Za-z]*` to split the lines, and we obtained the following output in a txt file with `hdfs dfs -getmerge /user/user_lsc_3/labPySparkData/output/* output.txt`

```
(43557, u'')
(22046, u'the')
(11676, u'and')
```

which are the most common words.