DISTRIBUTED SYSTEMS LABS -- WEEK5
NAME: Sagnik Chatterjee
REG: 180905478
ROLL NO : 61
Sec :B

Solved Examples:

1.Word Count Example

Mapper.py

```
import sys
# input comes from STDIN (standard input)
for line in sys.stdin:
   # remove leading and trailing whitespace
   line = line.strip()
   # split the line into words
   words = line.split()
   # increase counters
   for word in words:
       # write the results to STDOUT (standard output);
       # what we output here will be the input for the
       # Reduce step, i.e. the input for reducer.py
       #
       # tab-delimited; the trivial word count is 1
       print ('%s\t%s' % (word, 1))
```

Reducer.py

```
#!/usr/bin/env python
"""reducer.py"""
from operator import itemgetter
import sys
```

```
current_word = None
current_count = 0

word = None
# 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
   word, count = line.split('\t', 1)
   # 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
# this IF-switch only works because Hadoop sorts map output
# by key (here: word) before it is passed to the reducer
   if current_word == word:
       current_count += count
   else:
       if current_word:
           # write result to STDOUT
           print('%s\t%s' % (current_word, current_count))
           current_count = count
           current_word = word
           # do not forget to output the last word if needed!
if current_word == word:
   print('%s\t%s' % (current_word, current_count))
```

```
sagnik :: ~/ds_lab/week5final/samplePrograms >> echo "a a a a v v f f hh hh fg t
g fg gt nnn ccc ddd nnn ddd"|python mapper1.py
a       1
a       1
a       1
a       1
v       1
v       1
f       1
f       1
hh      1
hh      1
fg      1
tg      1
fg      1
gt      1
nnn     1
ccc     1
ddd     1
nnn     1
ddd     1
sagnik :: ~/ds_lab/week5final/samplePrograms >>
```

2. MapReduce program to find frequent words

Freqmap1.py

```
# as (word, 1)
from __future__ import print_function
import sys
# input comes from STDIN (standard input)
for line in sys.stdin:
   # create tuples of all words in line
   L = [(word.strip().lower(), 1) for word in
line.strip().split()]
   # increase counters
   for word, n in L:
       # write the results to STDOUT (standard output);
       # what we output here will be the input for the
       # Reduce step, i.e. the input for reducer.py
       #
       # tab-delimited; the trivial word count is 1
       print('%s\t%d' % (word, n))
```

Freqred1.py

```
#!/usr/bin/env python
# reducer.py
from __future__ import print_function
import sys
lastWord = None
sum = 0
for line in sys.stdin:
   word, count = line.strip().split('\t', 1)
   count = int(count)
   if lastWord == None:
       lastWord = word
       sum = count
       continue
   if word == lastWord:
       sum += count
   else:
       print("%s\t%d" % (lastWord, sum))
       sum = count
       lastWord = word
       # output last word
if lastWord == word:
   print('%s\t%s' % (lastWord, sum))
```

Freqmap2.py

```
#!/usr/bin/env python
# A basic mapper function/program that
# takes whatever is passed on the input and
# outputs tuples of all the words formatted
# as (word, 1)
from __future__ import print_function
import sys
```

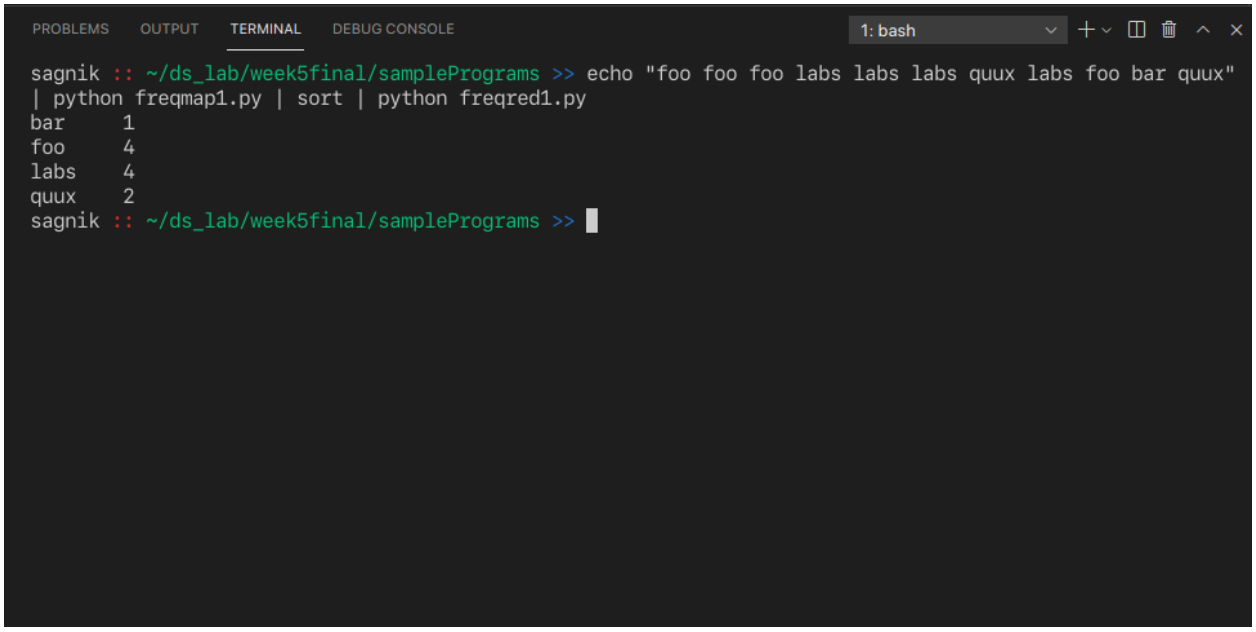
```
# input comes from STDIN (standard input)
for line in sys.stdin:
   word, count = line.strip().split('\t', 1)
   count = int(count)
   print('%d\t%s' % (count, word))
```

Freqred2.py

```
#!/usr/bin/env python
# reducer.py
from __future__ import print_function
import sys
mostFreq = []
currentMax = -1
for line in sys.stdin:
   count, word = line.strip().split('\t', 1)
   count = int(count)
   if count > currentMax:
       currentMax = count
       mostFreq = [word]
   elif count == currentMax:
       mostFreq.append(word)
       # output mostFreq word(s)
for word in mostFreq:
   print('%s\t%s' % (word, currentMax))
```

```
sagnik :: ~/ds_lab/week5final/samplePrograms >> echo "foo foo foo labs labs labs quux labs foo bar quux"
| python freqmap1.py | sort | python freqred1.py | python freqmap2.py | sort
1       bar
2       quux
4       foo
4       labs
sagnik :: ~/ds_lab/week5final/samplePrograms >>
```

```
sagnik :: ~/ds_lab/week5final/samplePrograms >> echo "foo foo foo labs labs labs quux labs foo bar quux"
| python freqmap1.py | sort | python freqred1.py | python freqmap2.py
1       bar
4       foo
4       labs
2       quux
sagnik :: ~/ds_lab/week5final/samplePrograms >>
```



3. MapReduce program to explore the dataset and perform the filtering (typically creating key/value pairs) by mapper and perform the count and summary operation on the instances.

Mapper.py

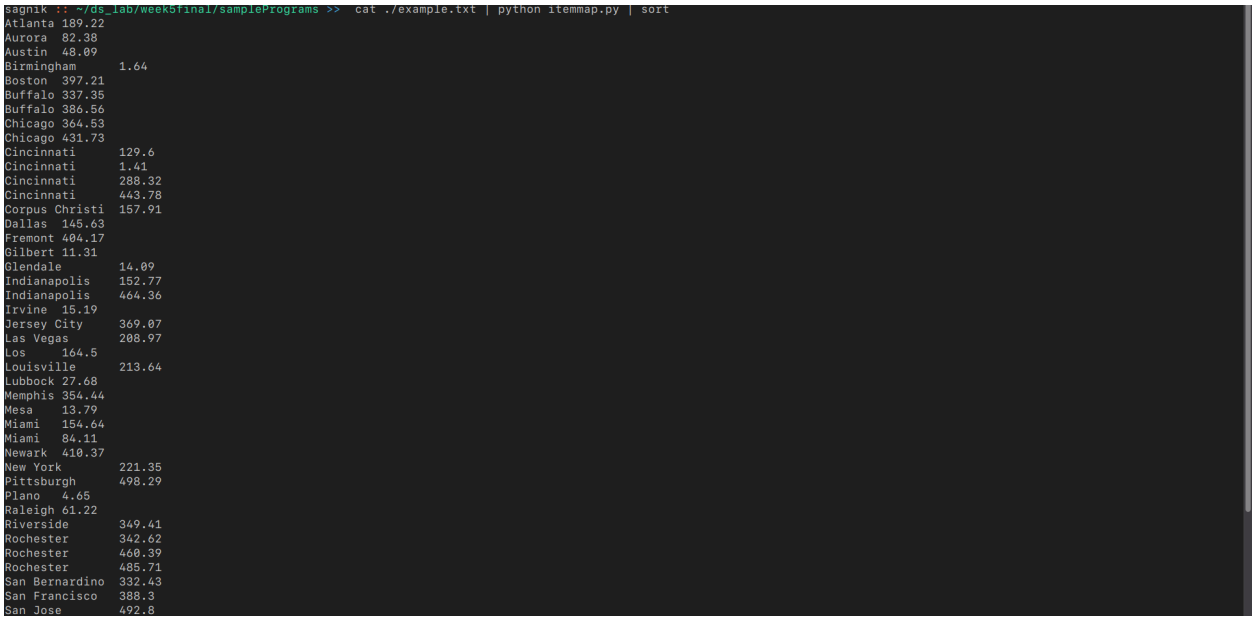
```
import fileinput
for line in fileinput.input():
   data = line.strip().split("\t")
   if len(data) == 6:
```

```
        date, time, location, item, cost, payment = data
        print("{0}\t{1}".format(location, cost))
        # can try with different instances.....
        #print ("{0}\t{1}".format(payment, cost))
        #print ("{0}\t{1}".format(item, cost))
```

Reducer.py

```
import fileinput
transactions_count = 0
sales_total = 0

for line in fileinput.input():
   data = line.strip().split("\t")
   if len(data) != 2:
       # Something has gone wrong. Skip this line.
       continue
   current_key, current_value = data
   transactions_count += 1
   sales_total += float(current_value)
print(transactions_count, "\t", sales_total)
```

```
sagnik :: ~/ds_lab/week5final/samplePrograms >> cat ./example.txt | python itemm
ap.py | sort | python itemread.py
50      12268.159999999996
```



```
sagnik :: ~/ds_lab/week5final/samplePrograms >>  cat ./example.txt | python itemmap.py | sort
Atlanta 189.22
Aurora  82.38
Austin  48.09
Birmingham      1.64
Boston  397.21
Buffalo 337.35
Buffalo 386.56
Chicago 364.53
Chicago 431.73
Cincinnati      129.6
Cincinnati      1.41
Cincinnati      288.32
Cincinnati      443.78
Corpus Christi  157.91
Dallas  145.63
Fremont 404.17
Gilbert 11.31
Glendale        14.09
Indianapolis    152.77
Indianapolis    464.36
Irvine  15.19
Jersey City     369.07
Las Vegas       208.97
Los     164.5
Louisville      213.64
Lubbock 27.68
Memphis 354.44
Mesa    13.79
Miami   154.64
Miami   84.11
Newark  410.37
New York        221.35
Pittsburgh      498.29
Plano   4.65
Raleigh 61.22
Riverside       349.41
Rochester       342.62
Rochester       460.39
Rochester       485.71
San Bernardino  332.43
San Francisco   388.3
San Jose        492.8
```

4. Write a mapper and reducer program for word count by defining separator instead of using “\t”

Mapper.py

```python
#!/usr/bin/env python
"""A more advanced Mapper, using Python iterators and
generators."""
import sys


def read_input(file):
    for line in file:
        # split the line into words
        yield line.split()


def main(separator='\t'):
    # input comes from STDIN (standard input)
```

```
    data = read_input(sys.stdin)
    for words in data:
        # write the results to STDOUT (standard output);
        # what we output here will be the input for the
        # Reduce step, i.e. the input for reducer.py
        # tab-delimited; the trivial word count is 1
        for word in words:
            print('%s%s%d' % (word, separator, 1))


if __name__ == "__main__":
    main()
```

Reducer.py

```
from itertools import groupby
from operator import itemgetter
import sys


def read_mapper_output(file, separator='\t'):
    for line in file:
        yield line.rstrip().split(separator, 1)


def main(separator='\t'):

    # input comes from STDIN (standard input)
    data = read_mapper_output(sys.stdin, separator=separator)
    # groupby groups multiple word-count pairs by word,
    # and creates an iterator that returns consecutive keys and
their group:
    # current_word - string containing a word (the key)
```

```
    # group - iterator yielding all ["&lt;current_word&gt;",
"&lt;count&gt;"] items
    for current_word, group in groupby(data, itemgetter(0)):
        try:
            total_count = sum(int(count) for current_word, count
in group)
            print("%s%s%d" % (current_word, separator,
total_count))
        except ValueError:
            # count was not a number, so silently discard this
item
            pass


if __name__ == "__main__":
    main()
```

```
sagnik :: ~/ds_lab/week5final/samplePrograms >> echo "TIme is gold is Time gold"
 | python sepmap.py | python spred.py | sort | python spred.py
gold    2
is      2
Time    1
TIme    1
sagnik :: ~/ds_lab/week5final/samplePrograms >>
```

```
sagnik :: ~/ds_lab/week5final/samplePrograms >> echo " Time is gold Time is Time
 gold" | python3 sepmap.py|python3 spred.py
Time    1
is      1
gold    1
Time    1
is      1
Time    1
gold    1
sagnik :: ~/ds_lab/week5final/samplePrograms >>
```

5. Write a map reduce program that returns the cost of the item that is most expensive, for each
location in the dataset example.txt

Mapper.py

```
import fileinput

for line in fileinput.input():
    data = line.strip().split("\t")
    if len(data) == 6:
        date, time, location, item, cost, payment = data
        print("{0}\t{1}".format(location, cost))
```

Reducer.py

```
import fileinput
max_value = 0
old_key = None

for line in fileinput.input():
    data = line.strip().split("\t")
    if len(data) != 2:
        # Something has gone wrong. Skip this line.
        continue
    current_key, current_value = data
    # Refresh for new keys (i.e. locations in the example
context)
    if old_key and old_key != current_key:
        print(old_key, "\t", max_value)
        old_key = current_key
        max_value = 0
        old_key = current_key
        if float(current_value) > float(max_value):
            max_value = float(current_value)
if old_key != None:
    print(old_key, "\t", max_value)
```

```
sagnik :: ~/ds_lab/week5final/samplePrograms >> cat ./example.txt | python itemmap_expensive.py | sort
Atlanta 189.22
Aurora  82.38
Austin  48.09
Birmingham      1.64
Boston  397.21
Buffalo 337.35
Buffalo 386.56
Chicago 364.53
Chicago 431.73
Cincinnati      129.6
Cincinnati      1.41
Cincinnati      288.32
Cincinnati      443.78
Corpus Christi  157.91
Dallas  145.63
Fremont 404.17
Gilbert 11.31
Glendale        14.09
```

6. Write a mapreduce program to evaluate the PI.

Mapper.py

```python
import sys



def f(x):
   return 4.0 / (1.0 + x*x)



# input comes from STDIN (standard input)
for line in sys.stdin:
   # remove leading and trailing whitespace
   line = line.strip()
   # split the line into words
   words = line.split()
   N = int(words[0])
   deltaX = 1.0 / N
   for i in range(0, N):
       print("1\t%1.10f" % (f(i * deltaX)*deltaX))
```

Reducer.py

```
#!/usr/bin/env python
from __future__ import print_function
from operator import itemgetter
import sys

sum1 = 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
   word, count = line.split('\t', 1)
   # convert count (currently a string) to int
   try:
       count = float(count)
   except ValueError:
       # count was not a number, so silently
       # ignore/discard this line
       #print( "--skipping (%s, %s)" % ( str(word), str(count) )
)
       continue
   sum1 += count
# do not forget to output the last word if needed!
print('%1.10f\t0' % sum1)
```

```
sagnik :: ~/ds_lab/week5final/samplePrograms >> echo "5" | python pi_mapper.py
1       0.8000000000
1       0.7692307692
1       0.6896551724
1       0.5882352941
1       0.4878048780
sagnik :: ~/ds_lab/week5final/samplePrograms >> echo "5" | python pi_mapper.py |
 python pi_reducer.py
  File "/home/sagnik/ds_lab/week5final/samplePrograms/pi_reducer.py", line 21
    continue
    ^
IndentationError: expected an indented block
sagnik :: ~/ds_lab/week5final/samplePrograms >> echo "5" | python pi_mapper.py |
 python pi_reducer.py
3.3349261137    0
sagnik :: ~/ds_lab/week5final/samplePrograms >> echo "3" | python pi_mapper.py
1       1.3333333333
1       1.2000000000
1       0.9230769231
sagnik :: ~/ds_lab/week5final/samplePrograms >>
```

Exercises

1. **Try the above word count program for the Heart Disease dataset, covid_19_data dataset, example dataset and German Credit dataset.Students can decide their own way of displaying results (can work on any columns in the dataset) on the dataset mentioned.**

   On the heart Deiseasse dataset
   Mapper.py

```python
import sys
import pandas as pd



df = pd.read_csv('heart_disease_data.csv')
age=df['age']
for word in age:
   print(word,'\t',1)
```

Reducer.py

```
from operator import itemgetter
import sys

current_word = None
current_count = 0
word = None

# input from stdin
for line in sys.stdin:
    line = line.strip()
    word, count = line.split('\t', 1)
    try:
        count = int(count)
    except ValueError:
        continue
    if current_word == word:
        current_count += count
    else:
        if current_word:
            print(f"{current_word} {current_count}")
        current_count = count
        current_word = word

if current_word == word:
    print(f"{current_word} {current_count}")
```

```
sagnik :: ~/ds_lab/week5final/q1 >>  python mapper.py | sort |python reducer.py
29  1
34  2
35  4
37  2
38  3
39  4
40  3
41  10
42  8
43  8
44  11
45  8
46  7
47  5
48  7
49  5
50  7
51  12
52  13
53  8
54  16
55  8
56  11
57  17
58  19
59  14
60  11
61  8
62  11
63  9
64  10
65  8
66  7
67  9
68  4
69  3
70  4
71  3
74  1
76  1
77  1
```

2. **Try the above frequent word count program for the Heart Disease dataset,**
   covid_19_data dataset, example dataset and German Credit data.
   Students can decide their own way of displaying results (can work on any columns in the
   dataset) on the dataset mentioned.

Freqmap1.py

```
from __future__ import print_function
import sys
import pandas as pd

df = pd.read_csv('covid_19_data.csv')
country = df['Deaths']
for word in country:
   print(word, '\t', 1)
```

Freqmap2.py

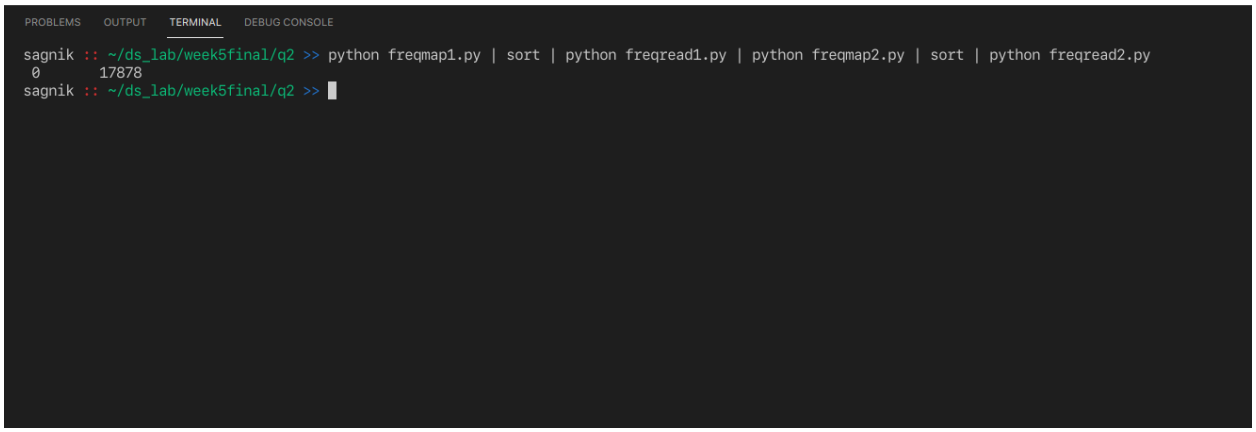
```
from __future__ import print_function
import sys

for line in sys.stdin:
   word,count = line.strip().split('\t',1)
   count = int(count)
   print(count,'\t',word)
```

Freqread1.py

```
from __future__ import print_function
import sys
lastWord = None

sum = 0

for line in sys.stdin:
   word,count = line.strip().split('\t',1)
   count = int(count)
   if lastWord == None:
       lastWord = word
       sum = count
       continue
   if word == lastWord:
       sum += count
   else:
       print(lastWord,'\t',sum)
       sum = count
       lastWord = word

#output the last word
if lastWord == word:
   print(lastWord,'\t',sum)
```

Freqread2.py

```
from __future__ import print_function

import sys

mostFreq = []
currentMax = -1
for line in sys.stdin:
   count,word = line.strip().split('\t',1)
   count = int(count)
   if count > currentMax :
       currentMax = count
       mostFreq = [word]
   elif count == currentMax:
       mostFreq.append(word)
#output the mostFreq word(s)
for word in mostFreq:
   print(word,'\t',currentMax)
```



3. **Try the above ‘Item explore and count program’ for the Heart Disease dataset,**
   covid_19_data dataset, example dataset and German Credit dataset.
   Students can decide their own way of displaying results (can work on any columns in the
   dataset) on the dataset mentioned.

Mapper.py

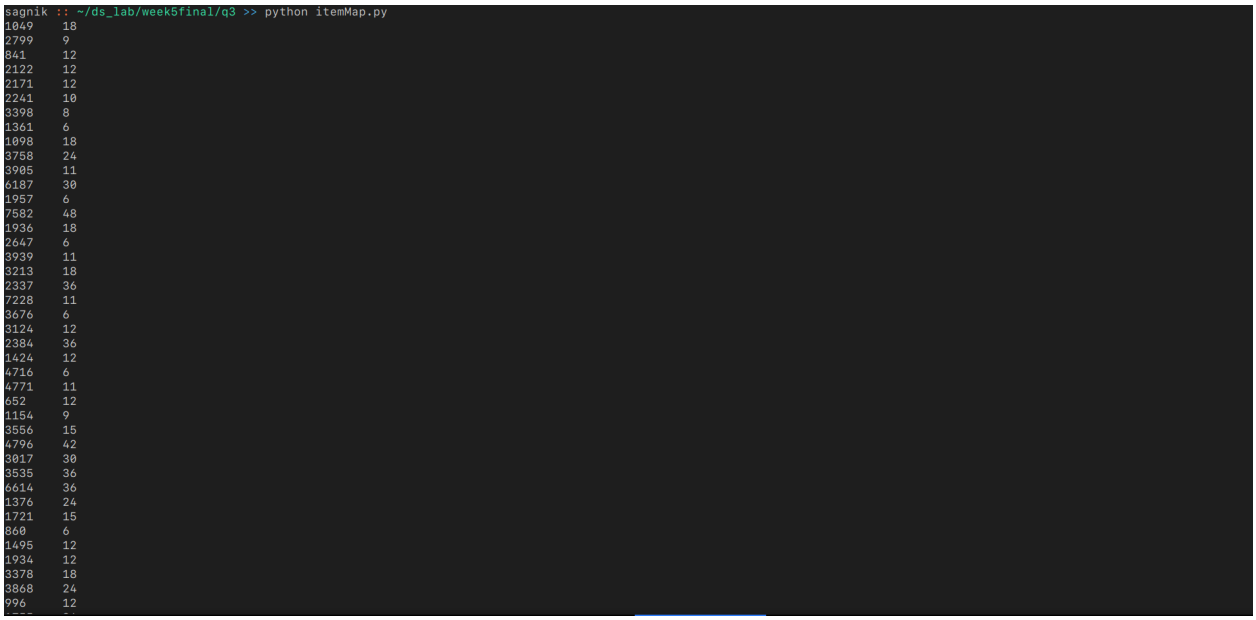
```
import pandas as pd
import numpy as np

dataFrame  = pd.read_excel('German_Credit.xlsx')
x = dataFrame["CreditAmount"]
y = dataFrame["DurationOfCreditInMonths"]

for i in range(len(x)):
   print(f"{x[i]}\t{y[i]}")
```

Reducer.py

```
import fileinput
transactions_count = 0

sales_total = 0
for line in fileinput.input():
   data= line.strip().split("\t")
   if len(data)!=2:
       #skip line
       continue
   current_key,current_value=data
   transactions_count +=1
   sales_total+=float(current_value)
```

```
print(transactions_count,"\t",sales_total)
```

```
sagnik :: ~/ds_lab/week5final/q3 >> python itemMap.py | sort | python itemReduc
er.py
1000     20903.0
sagnik :: ~/ds_lab/week5final/q3 >>
```



4. **Try to include separator using map reducing for the output of Heart Disease dataset, covid_19_data dataset, example dataset and German Credit dataset**

Mapper.py

```
import pandas as pd
import numpy as np

def dateframe_input(dataframe):
    for x in dataframe['age']:
        yield x
```

```
def main(separator='\t'):
    dataFrame = pd.read_csv('heart_disease_data.csv')
    data = dateframe_input(dataFrame)
    for x in data:
        print('%s%s%d' %(x,separator,1))

if __name__=='__main__':
    main()
```

Reducer.py

```
from itertools import groupby
from operator import itemgetter

import sys

def read_file_input(file,separator="\t"):
    for line in file:
        yield line.rstrip().split(separator,1)

def main(separator='\t'):
    data = read_file_input(sys.stdin,separator=separator)
    for current_word,group in groupby(data,itemgetter(0)):
        try:
            total_count = sum(int(count) for
current_word,count in group)
            print("%s--> %s\t%d"
%(current_word,separator,total_count))
        except Exception as e:
            print(e)

if __name__ =='__main__':
    main()
```

```
agnik :: ~/ds_lab/week5final/q4 >> python seperatorMap.py | sort | python seperatorReduce.py
29-->           1
34-->           2
35-->           4
37-->           2
38-->           3
39-->           4
40-->           3
41-->           10
42-->           8
43-->           8
44-->           11
45-->           8
46-->           7
47-->           5
48-->           7
49-->           5
50-->           7
51-->           12
52-->           13
53-->           8
54-->           16
55-->           8
56-->           11
57-->           17
58-->           19
59-->           14
60-->           11
61-->           8
62-->           11
63-->           9
64-->           10
65-->           8
66-->           7
67-->           9
68-->           4
69-->           3
70-->           4
71-->           3
74-->           1
76-->           1
77-->           1
```

5. **Try to apply finding max value using map reduce concept for the output of Heart Disease dataset, covid_19_data dataset, example dataset and German Credit dataset. Students can decide their own way of displaying results (can work on any columns in the dataset) on the dataset mentioned**

Mapper.py

```
import numpy as np
import pandas as pd

dataframe = pd.read_csv("covid_19_data.csv")
country_name = dataframe["Country/Region"]
peak_cases = dataframe["Confirmed"]

for x in range(len(country_name)):
   print("%s\t%d" %(country_name[x],peak_cases[x]))
```

Reducer.py

```
import fileinput
max_value = -1
```

```
old_key = None

for line in fileinput.input():
    data = line.strip().split("\t")
    if len(data) == 2:
        current_key, current_value = data
        if old_key == current_key:
            if float(current_value) > float(max_value):
                max_value = float(current_value)
        else:
            if old_key != None:
                print(old_key, "-->", max_value, end="\n")
            old_key = current_key
            max_value = current_value
```

```
Ethiopia --> 71083.0
Faroe Islands --> 2.0
Fiji --> 32.0
Finland --> 9288.0
France --> 483956.0
French Guiana --> 18.0
Gabon --> 8716.0
Gambia --> 3542.0
Gambia, The --> 1
Georgia --> 4140.0
Germany --> 174098.0
Ghana --> 46153.0
Gibraltar --> 1
Greece --> 16286.0
Greenland --> 1.0
Grenada --> 24.0
Guadeloupe --> 53.0
Guam --> 3.0
Guatemala --> 87442.0
Guernsey --> 1.0
Guinea --> 10434.0
Guinea-Bissau --> 2324.0
Guyana --> 2535.0
Haiti --> 8646.0
Holy See --> 12.0
Honduras --> 72675.0
Hong Kong --> 5049.0
Hungary --> 20450.0
Iceland --> 2476.0
India --> 1242770.0
Indonesia --> 257388.0
Iran --> 432798.0
Iraq --> 332635.0
Ireland --> 33675.0
Israel --> 204690.0
Italy --> 222104.0
Ivory Coast --> 19430.0
Jamaica --> 5395.0
Japan --> 24463.0
Jersey --> 2.0
Jordan --> 6042.0
Kazakhstan --> 107529.0
Kenya --> 37348.0
Kosovo --> 12683.0
Kuwait --> 101299.0
```

```
sagnik :: ~/ds_lab/week5final/q5 >> python mapper.py | sort | python reducer.py
Afghanistan --> 39145.0
Albania --> 12787.0
Algeria --> 50400.0
Andorra --> 1753.0
Angola --> 4363.0
Antigua and Barbuda --> 97.0
Argentina --> 664799.0
Armenia --> 47877.0
Aruba --> 4.0
Australia --> 20105.0
Austria --> 39984.0
Azerbaijan --> 39524.0
Bahamas --> 3618.0
Bahamas, The --> 4.0
Bahrain --> 67014.0
Bangladesh --> 353844.0
Barbados --> 189.0
Belarus --> 76357.0
Belgium --> 106887.0
Belize --> 1706.0
Benin --> 2325.0
Bhutan --> 261.0
Bolivia --> 131990.0
Bosnia and Herzegovina --> 26081.0
Botswana --> 2567.0
Brazil --> 945422.0
Brunei --> 145.0
Bulgaria --> 19283.0
Burkina Faso --> 1929.0
Burma --> 7827.0
Burundi --> 476.0
Cabo Verde --> 5412.0
Cambodia --> 275.0
Cameroon --> 20690.0
Canada --> 69088.0
Cape Verde --> 1
Cayman Islands --> 1
Central African Republic --> 4802.0
Chad --> 1164.0
Channel Islands --> 1
Chile --> 283748.0
Colombia --> 257679.0
Comoros --> 470.0
Congo (Brazzaville) --> 5005.0
```

7. Write a MapReduce program to count even or odd numbers in randomly generated natural numbers

randGen.py

```
import random
```

```
for i in range(1000):
   n = random.randint(1,1000)
   print(n)
```

Mapper.py

```
import sys

for line in sys.stdin:
   line = line.strip()
   nums = line.split()
   for num in nums:
       val = int(num)
       if val % 2 == 0:
           print('%s\t%s' %("EVEN COUNT",1))
       else:
           print('%s\t%s' %("ODD  COUNT",1))
```

Reducer.py

```
from operator import itemgetter
import sys

current_word = None
current_count = 0
word = None

for line in sys.stdin:
   line = line.strip()
   word, count = line.split('\t', 1)

   try:
       count = int(count)
   except ValueError as e:
       # case when count NaN , debug this and continue
```

```
        print(e)
        continue

    if current_word == word:
        current_count += 1
    else:
        if current_word:
            print('%s\t%s' % (current_word, current_count))

        current_count = count
        current_word = word

if current_word == word:
    print(f"{current_word}  {current_count}")
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
sagnik :: ~/ds_lab/week5final/q7 >> ls
mapper.py  randGen.py  reducer.py
sagnik :: ~/ds_lab/week5final/q7 >> python randGen.py | python mapper.py | sort | python reducer.py
EVEN COUNT      489
ODD  COUNT  511
sagnik :: ~/ds_lab/week5final/q7 >> python randGen.py | python mapper.py | sort | python reducer.py
EVEN COUNT      501
ODD  COUNT  499
sagnik :: ~/ds_lab/week5final/q7 >> python randGen.py | python mapper.py | sort | python reducer.py
EVEN COUNT      518
ODD  COUNT  482
sagnik :: ~/ds_lab/week5final/q7 >>
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
sagnik :: ~/ds_lab/week5final/q7 >> ls
mapper.py  randGen.py  reducer.py
sagnik :: ~/ds_lab/week5final/q7 >> python randGen.py | python mapper.py | sort | python reducer.py
EVEN COUNT      489
ODD  COUNT  511
sagnik :: ~/ds_lab/week5final/q7 >>
```

6. **Write a MapReduce program to generate a report with Number of males, females**
   and total births in each year, number of males, females and total births in each month of a
   particular year from national birth data.

Mapper.py

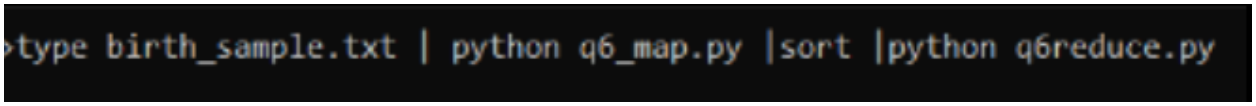
```
import sys
for line in sys.stdin:
   line = line.strip()
   vals = line.split()
   print('%s %s %s %s %s' % (vals[14], vals[15], vals[16],
vals[31], 1))
```

Reducer.py

```
import sys
from operator import itemgetter
n_males = 0
n_females = 0
curr_year = 0
year = None
t_females = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
t_males = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
print("Yearly Statistics")
for line in sys.stdin:
   line = line.strip()
   year, month, date, gender, count = line.split(' ', 4)
   try:
       count = int(count)
   except ValueError:
       continue

   if gender == 'M':
       t_males[int(month)] += 1
```

```
    else:
        t_females[int(month)] += 1
    if int(year) == curr_year:
        if gender == 'M':
            n_males += 1
        else:
            n_females += 1
    else:
        if curr_year >= 0:
            print('year %s females: %s males: %s total
births: %s' %
                  (curr_year, n_females, n_males,
n_males+n_females))
    curr_year = int(year)
    if gender == 'M':
        n_males = 1
    else:
        n_females = 1
if curr_year == year:
    print('year %s females: %s males: %s total births: %s' %
          (curr_year, n_females, n_males,
n_males+n_females))
print("")
print("Monthly Statistics")
for i in range(0, 12):
    print('month %s males: %s females: %s total births: %s'
%
          (i+1, t_males[i], t_females[i],
t_males[i]+t_females[i]))
```

Screenshot:



```
Yearly Statistics
year 0 females: 1 males: 1 total births: 2
year 1 females: 1 males: 1 total births: 2
year 2 females: 1 males: 1 total births: 2
year 3 females: 1 males: 1 total births: 2
year 4 females: 1 males: 1 total births: 2
year 5 females: 1 males: 1 total births: 2
year 6 females: 1 males: 1 total births: 2
year 7 females: 1 males: 1 total births: 2
year 8 females: 1 males: 1 total births: 2
year 9 females: 1 males: 1 total births: 2
year 10 females: 1 males: 1 total births: 2
year 11 females: 1 males: 1 total births: 2
year 12 females: 1 males: 1 total births: 2
year 13 females: 1 males: 1 total births: 2
year 14 females: 1 males: 1 total births: 2
year 16 females: 1 males: 1 total births: 2
year 20 females: 1 males: 1 total births: 2
year 21 females: 1 males: 1 total births: 2

Monthly Statistics
month 1 males: 0 females: 4 total births: 4
month 2 males: 20801 females: 19806 total births: 40607
month 3 males: 16215 females: 15416 total births: 31631
month 4 males: 8439 females: 8011 total births: 16450
month 5 males: 3421 females: 3280 total births: 6701
month 6 males: 1319 females: 1202 total births: 2521
month 7 males: 529 females: 511 total births: 1040
month 8 males: 244 females: 231 total births: 475
month 9 males: 234 females: 211 total births: 445
month 10 males: 72 females: 54 total births: 126
month 11 males: 0 females: 0 total births: 0
month 12 males: 0 females: 0 total births: 0
```