

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

```
# sepmap.py
# 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()

# sepred.py
# more advanced Reducer, using Python iterators and generators
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 ["<current_word>", "<count>"] 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()
```

```
210905189_Kushala@networklab:~/Desktop/210905189/Lab6/solved$ echo "Time is gold Time is Time gold" | python3 sepmap.py
Time 1
is 1
gold 1
Time 1
is 1
Time 1
gold 1
210905189_Kushala@networklab:~/Desktop/210905189/Lab6/solved$

210905189_Kushala@networklab:~/Desktop/210905189/Lab6/solved$ echo " Time is gold Time is Time gold" | python3 sepmap.py|sort|python3 sepred.py
gold 2
is 2
Time 3
210905189_Kushala@networklab:~/Desktop/210905189/Lab6/solved$
```

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

```
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))
```

```
import fileinput
```

The image shows a Linux terminal window with a dark purple background. The title bar at the top indicates the date and time as 'Feb 26 14:18'. The terminal displays a list of cities and their corresponding values, organized into two columns. The cities are listed in descending order of value. The terminal prompt is '210905189_Kushala@networklab: ~/Desktop/210905189/Lab6/solved\$'.

City	Value
Newark	410.37
Memphis	354.44
Jersey City	369.07
Plano	4.65
Buffalo	337.35
Louisville	213.64
Miami	154.64
Los	164.5
Birmingham	1.64
Mesa	13.79
Wichita	158.25
Indianapolis	152.77
San Bernardino	332.43
Indianapolis	464.36
Stockton	180.61
Austin	48.09
Buffalo	386.56
Santa Ana	2.75
Gilbert	11.31
New York	221.35
Corpus Christi	157.91
Riverside	349.41
Chicago	364.53
Fremont	404.17
Rochester	460.39
Raleigh	61.22
Chicago	431.73
Cincinnati	288.32
Rochester	342.62
Pittsburgh	498.29
Rochester	485.71
Glendale	14.09
Cincinnati	1.41
Irvine	15.19
Boston	397.21
Scottsdale	214.32
Atlanta	189.22
Cincinnati	443.78
Lubbock	27.68
Cincinnati	129.6
Santa Ana	282.13
Aurora	82.38

```
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)
        max_value = 0

    if float(current_value) > float(max_value):
        max_value = float(current_value)

    old_key = current_key

if old_key is not None:
    print(old_key, "\t", max_value)

```

```

210905189_Kushala@networklab: ~/Desktop/210905189/Lab6/solved
210905189_Kushala@networklab: ~/Desktop/210905189/Lab6/solved$ cat example.txt | python3 itemmap_expensive.py | sort | python3 itemred_expensive.py
Time 3
210905189_Kushala@networklab: ~/Desktop/210905189/Lab6/solved$
Atlanta 189.22
Aurora 82.38
Austin 48.09
Birmingham 1.64
Boston 397.21
Buffalo 386.56
Chicago 431.73
Cincinnati 443.78
Corpus Christi 157.91
Dallas 145.63
Frenont 484.17
Gilbert 11.31
Glendale 14.09
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
Newark 410.37
New York 221.35
Pittsburgh 498.29
Plano 4.65
Raleigh 61.22
Riverside 349.41
Rochester 485.71
San Bernardino 332.43
San Francisco 380.3
San Jose 492.8
Santa Ana 282.13
Scottsdale 214.32
Stockton 180.61
Tampa 353.23
Tucson 489.93
Washington 481.31
Wichita 158.25
210905189_Kushala@networklab: ~/Desktop/210905189/Lab6/solved$

```

6. Write a mapreduce program to evaluate the PI.

```
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))

```

```

from __future__ import print_function
import sys

```

```

sum = 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 float
    try:
        count = float(count)
    except ValueError:
        # count was not a number, so silently
        # ignore/discard this line
        continue
    sum += count

```

```

# print the total sum
print("%1.10f\t0" % sum)

```

```

210905189_Kushala@networklab:~/Desktop/210905189/Lab6/solved$ echo "5"|python3 mapper_pi.py|python3 reducer_pi.py
3.3349261137    0
210905189_Kushala@networklab:~/Desktop/210905189/Lab6/solved$

```

```

210905189_Kushala@networklab:~/Desktop/210905189/Lab6/solved$ echo "5"|python3 mapper_pi.py
1      0.8000000000
1      0.7692307692
1      0.6896551724
1      0.5882352941
1      0.4878048780

```