

Part 1a:

$$2^{12} = 2^{12} = 4096$$

$$4^5 = 4^5 = 1024$$

$$4^5 = 2^{(2*5)} = 1024$$

1221 MOD 12 (MOD is the modulo operator, a.k.a. the remainder)

Ans: Remainder is 9

61 MOD 13

Ans: Remainder is 9

43 MOD 7

Ans: Remainder is 1

Part 1b:

V1 = (2, 1, 3) and V2 = (1, 1, 2) and a 3x3 matrix M = [(1, 1, 1), (2, 1, 3), (1, 0, 2)]

V1 + V2 (vector addition)

Ans: (3, 2, 5)

V1 - V2 (vector subtraction)

Ans: (1, 0, 1)

|V1| (vector length)

Ans:  $\sqrt{2^2 + 1^2 + 3^2} = \sqrt{14} = 3.7416573867739413$

(Python Version: `np.sqrt((v1**2).sum()) = 3.7416573867739413`)

|V2| (vector length)

Ans:  $\sqrt{1^2 + 1^2 + 2^2} = \sqrt{6} = 2.4494897427831779$

(Python Version: `np.sqrt((v2**2).sum()) = 2.4494897427831779`)

M \* V1 (matrix times vector)

Ans: M \* Transpose (V1) = Matrix (3x3) \* V1.T (3x1) = Matrix (3x1) =  $\begin{pmatrix} 6 \\ 14 \\ 8 \end{pmatrix}$

Python Version:

```
In [13]: ##M * V1 (matrix times vector)
          M * v1.T
Out[13]: matrix([[ 6],
                  [14],
                  [ 8]])
```

$M * M$  (or  $M^2$ )

Ans:  $M * M = \text{Matrix}(3 \times 3) * \text{Matrix}(3 \times 3) = \text{Matrix}(3 \times 3) = \begin{pmatrix} 4 & 2 & 6 \\ 7 & 3 & 11 \\ 3 & 1 & 5 \end{pmatrix}$

Python Version:

```
In [15]: ##M * M (or M^2)
M**2

Out[15]: matrix([[ 4,  2,  6],
                 [ 7,  3, 11],
                 [ 3,  1,  5]])
```

$M^4$

Ans:  $M * M * M * M = \text{Matrix}(3 \times 3) * \text{Matrix}(3 \times 3) * \text{Matrix}(3 \times 3) * \text{Matrix}(3 \times 3)$   
 $= \text{Matrix}(3 \times 3) = \begin{pmatrix} 4 & 2 & 6 \\ 48 & 20 & 76 \\ 82 & 34 & 130 \\ 34 & 14 & 54 \end{pmatrix}$

Python Version:

```
In [16]: ##M^4
M**4

Out[16]: matrix([[ 48,  20,  76],
                 [ 82,  34, 130],
                 [ 34,  14,  54]])
```

Part 1c:  $\text{Pro}(H)=0.6$  And  $\text{Pro}(T)=0.4$

HHT

Ans:  $\text{Probability}(\text{HHT}) = \text{Pro}(H) * \text{Pro}(H) * \text{Pro}(T) = 0.6 * 0.6 * 0.4 = 0.144$

THHT

Ans:  $\text{Probability}(\text{THHT}) = \text{Pro}(T) * \text{Pro}(H) * \text{Pro}(H) * \text{Pro}(T) = 0.4 * 0.6 * 0.6 * 0.4 = 0.0576$

Exactly 2 Heads out of a sequence of 3 coin flips.

Ans: Probability of Exactly 2 Heads out of a sequence of 3 coin flips. The followings are 8 possible events

- 1) HHH
- 2) HHT
- 3) HTH
- 4) THH
- 5) HTT
- 6) THT
- 7) TTH
- 8) TTT

Ans:  $C_2^3 * 0.6^2 * 0.4^1 = 0.432$

Exactly 1 Tail out of sequence of 3 coin flips.

Ans: Probability of Exactly 1 Tail out of a sequence of 3 coin flips. The followings are 8 possible events

- 1)HHH
- 2)HHT
- 3)HTH
- 4)THH
- 5)HTT
- 6)THT
- 7)TTH
- 8)TTT

Ans:  $C_1^3 * 0.4^1 * 0.6^2 = 0.432$  #Same as the above case with Exactly 2 heads in 3 coin flips.

Part 1d:

Table1: Employee(ID, Name, Address, Position)

Table2: Certificates(EID, CertName, Date1)

- i. Find all employees first name is “Jane” (assume that Name is the full name of employee and that there is no FirstName column)

Ans:

```
--Assume the Name = "FirstName LastName" :  
Select Name  
from Employee  
where upper(name) LIKE 'JANE%';
```

- ii. Find out how many different certifications (CertName) are stored in the database (you should not count the same CertName twice).

Ans:

```
Select count(unique(CertName)) As No_type_Cert  
from Certificates;
```

- iii. Find all employees that have fewer than 4 certifications (note that this should include 0 to be correct)

Ans:

```
Select id, name, address, position, count(Certificates.CERTNAME) AS No_Cert  
from Employee left outer join Certificates  
on Employee.ID = Certificates.EID  
group by id,name,address,position  
having count(Certificates.CERTNAME)<4;
```

Part 1e:

Mining of Massive Datasets, Exercise 1.3.3 : Suppose hash-keys are drawn from the population of all nonnegative integers that are multiples of some constant  $c$ , and hash function  $h(x)$  is  $x \bmod 15$ . For what **values** of  $c$  will  $h$  be a suitable hash function, i.e., a large random choice of hash-keys will be divided roughly equally into buckets?

Ans:

Given:  $h(x) = x \bmod 15$  which values of  $x$  is multiple of  $c$

In order to send all the non-negative integers are sent to each of the 15 buckets which are 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14. I would suggest that the values of  $c$  would be 1,2,7,11,13,17 and these random drawn numbers with the above multiples would be sent to EACH of the 15 buckets. Please check the following tables to support this assumed result. Also, I have skipped to state the some constant values, e.g. 4,8, because they have factors of 2 and 4. Actually, the constant values of  $c$  could be more the above 6 values and those values are all highly possible prime numbers except the value of 1. Please refer to the follow tables refer to the values of  $c$  and the buckets.

Multiple values	c	value of x	B	Bucket
0	1	0	15	0
15	1	15		0
1	1	1		1
16	1	16		1
2	1	2		2
17	1	17		2
3	1	3		3
18	1	18		3
4	1	4		4
19	1	19		4
5	1	5		5
20	1	20		5
6	1	6		6
21	1	21		6
7	1	7		7
22	1	22		7
8	1	8		8
23	1	23		8
9	1	9		9
10	1	10		10
11	1	11		11
12	1	12		12
13	1	13		13
14	1	14		14

Multiple values	c	value of x	B	Bucket
0	2	0	15	0
15	2	30		0
8	2	16		1
23	2	46		1
1	2	2		2
16	2	32		2
9	2	18		3
2	2	4		4
17	2	34		4
10	2	20		5
3	2	6		6
18	2	36		6
11	2	22		7
4	2	8		8
19	2	38		8
12	2	24		9
5	2	10		10
20	2	40		10
13	2	26		11
6	2	12		12
21	2	42		12
14	2	28		13
7	2	14		14
22	2	44		14

Multiple values	c	value of x	B	Bucket
0	7	0	15	0
15	7	105		0
13	7	91		1
11	7	77		2
9	7	63		3
7	7	49		4
22	7	154		4
5	7	35		5
20	7	140		5
3	7	21		6
18	7	126		6
1	7	7		7
16	7	112		7
14	7	98		8
12	7	84		9
10	7	70		10
8	7	56		11
23	7	161		11
6	7	42		12
21	7	147		12
4	7	28		13
19	7	133		13
2	7	14		14
17	7	119		14

Multiple values	c	value of x	B	Bucket
0	11	0	15	0
15	11	165		0
11	11	121		1
7	11	77		2
22	11	242		2
3	11	33		3
18	11	198		3
14	11	154		4
10	11	110		5
6	11	66		6
21	11	231		6
2	11	22		7
17	11	187		7
13	11	143		8
9	11	99		9
5	11	55		10
20	11	220		10
1	11	11		11
16	11	176		11
12	11	132		12
8	11	88		13
23	11	253		13
4	11	44		14
19	11	209		14

Multiple values	c	value of x	B	Bucket	Multiple values	c	value of x	B	Bucket
0	13	0	15	0	0	17	0	15	0
15	13	195		0	15	17	255		0
11	13	143		8	8	17	136		1
7	13	91		1	23	17	391		1
22	13	286		1	1	17	17		2
3	13	39		9	16	17	272		2
18	13	234		9	9	17	153		3
14	13	182		2	2	17	34		4
10	13	130		10	17	17	289		4
6	13	78		3	10	17	170		5
21	13	273		3	3	17	51		6
2	13	26		11	18	17	306		6
17	13	221		11	11	17	187		7
13	13	169		4	4	17	68		8
9	13	117		12	19	17	323		8
5	13	65		5	12	17	204		9
20	13	260		5	5	17	85		10
1	13	13		13	20	17	340		10
16	13	208		13	13	17	221		11
12	13	156		6	6	17	102		12
8	13	104		14	21	17	357		12
23	13	299		14	14	17	238		13
4	13	52		7	7	17	119		14
19	13	247		7	22	17	374		14

Part 1f:

Generally speaking, The Mapper part is the code to read through the blocks of data, and then every word/key give value of 1

The Reducer phase is actually to use the function to aggregate Key-Value pair. The user would be in the role to decide what function to be applied. Avg, sum, std, count, min,max etc in this phase.

- i. Find the smallest number in the input file.

During the Map phase, the input file is split into different blocks. In the block, the Mapper function maps the data in the <key , value> format. The Mapper won't count or search for duplicates, so that the output from mapper would be some like <100, 1>, <200, 1> , <100,1>format.

And then the reducer takes the key-value pair from the Map phase, the reducer function aggregates the key and its value to a format <key, value>, e.g. <100, 2>, <200,10> which the values show the occurrence of the key within the block. After that the code should find out the smallest number (key) among all the numbers within the block.

- ii. Find all unique numbers that greater than 100 but less than 1000.

That Map phase is very similar to part i. During the Map phase, the input file is split into different blocks. In the block, the Mapper function maps the data in the <key , value> format. The Mapper won't count or search for duplicates, so that the output from mapper would be some like <100, 1>, <200, 1> , <3000,1>format.

And then the reducer takes the key-value pair from the Map phase, the reducer function aggregates the key and its value to a format <key, value>, e.g. <100, 2>, <200,10> which the values show the occurrence of the key within the block. After that the code should find out the number between 100 and 1000 (i.e. Python: if key <1000 and key>100) among all the number (key) within the block.

- iii. For a data file that contains records (ID, First, Last, Grade) for each student, find how many students with repeating names there are for each name entry.

In the Map phase, we need to map out all the keys which is First, Last from the student records. The query / aggregate SQL would be as the following

Map phase:

Select First, Last, ?  
from student  
group by First, Last

In the reduce phase, the reducer function will aggregate the keys (First, Last) from the map phase. The reducer function would be count(ID) refer to the ? on the Map phase Query.

- iv. For a data file that contains records (ID, First, Last, Grade) for each student, find the GPA of each student.

Assuming the Grade is in 4.0 Scale.

In the Map phase, the map function would map out the Keys <ID,First,Last> and Values <Grade> from the student record. The key-value pair format would be as <ID,First,Last, Grade>. The query/aggregate SQL would be as the following:

Map phase:

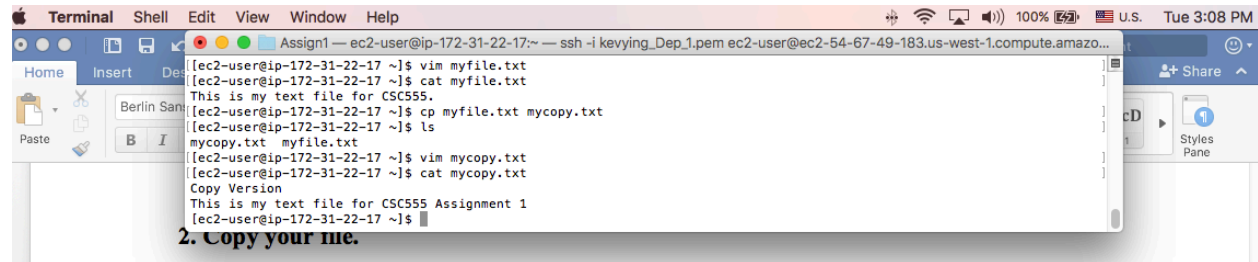
Select ID,First,Last, ?  
From Student  
Group by ID,First,Last

In the reduce phase, the reducer function will take the output from Map phase and compute the GPA for each Key < ID,First,Last > by using the function AVG(Grade) which refer to the ? in the Map phase Query.



Part 2:

**SUBMIT:** Take a screen shot of the contents of your copied file displayed on the terminal screen.

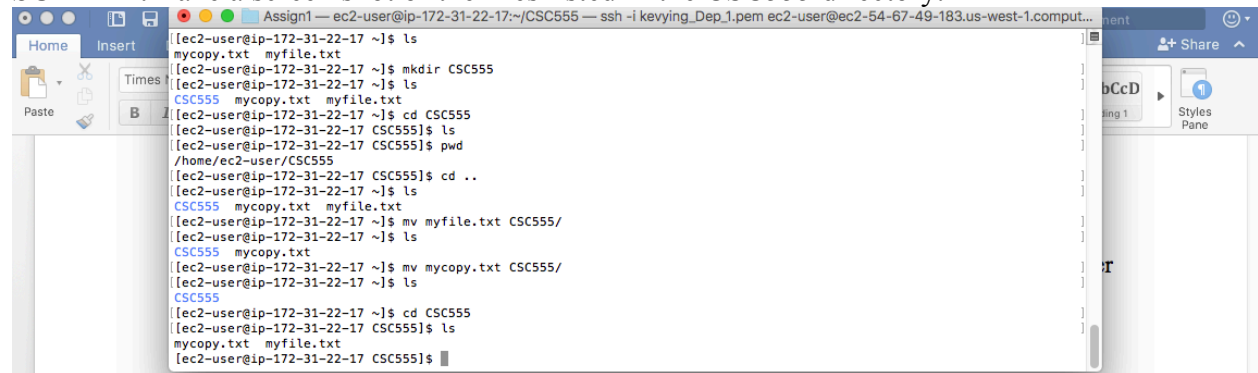


A terminal window titled 'Assign1' showing a user performing several commands. The user creates a file 'myfile.txt' with 'vim', displays its contents with 'cat', copies it to 'mycopy.txt' with 'cp', and lists the files with 'ls'. The terminal output is as follows:

```
ec2-user@ip-172-31-22-17 ~]$ vim myfile.txt
ec2-user@ip-172-31-22-17 ~]$ cat myfile.txt
This is my text file for CSC555.
ec2-user@ip-172-31-22-17 ~]$ cp myfile.txt mycopy.txt
ec2-user@ip-172-31-22-17 ~]$ ls
mycopy.txt  myfile.txt
ec2-user@ip-172-31-22-17 ~]$ vim mycopy.txt
ec2-user@ip-172-31-22-17 ~]$ cat mycopy.txt
Copy Version
This is my text file for CSC555 Assignment 1
ec2-user@ip-172-31-22-17 ~]$
```

Below the terminal window, the text "2. Copy your me." is visible.

**SUBMIT:** Take a screen shot of the files listed in the CSC555 directory.



A terminal window titled 'Assign1' showing a user creating a directory 'CSC555', moving files into it, and listing the contents. The terminal output is as follows:

```
ec2-user@ip-172-31-22-17 ~]$ ls
mycopy.txt  myfile.txt
ec2-user@ip-172-31-22-17 ~]$ mkdir CSC555
ec2-user@ip-172-31-22-17 ~]$ ls
CSC555  mycopy.txt  myfile.txt
ec2-user@ip-172-31-22-17 ~]$ cd CSC555
ec2-user@ip-172-31-22-17 CSC555]$ ls
ec2-user@ip-172-31-22-17 CSC555]$ pwd
/home/ec2-user/CSC555
ec2-user@ip-172-31-22-17 CSC555]$ cd ..
ec2-user@ip-172-31-22-17 ~]$ ls
CSC555  mycopy.txt  myfile.txt
ec2-user@ip-172-31-22-17 ~]$ mv myfile.txt CSC555/
ec2-user@ip-172-31-22-17 ~]$ ls
CSC555  mycopy.txt
ec2-user@ip-172-31-22-17 ~]$ mv mycopy.txt CSC555/
ec2-user@ip-172-31-22-17 ~]$ ls
CSC555
ec2-user@ip-172-31-22-17 ~]$ cd CSC555
ec2-user@ip-172-31-22-17 CSC555]$ ls
mycopy.txt  myfile.txt
ec2-user@ip-172-31-22-17 CSC555]$
```

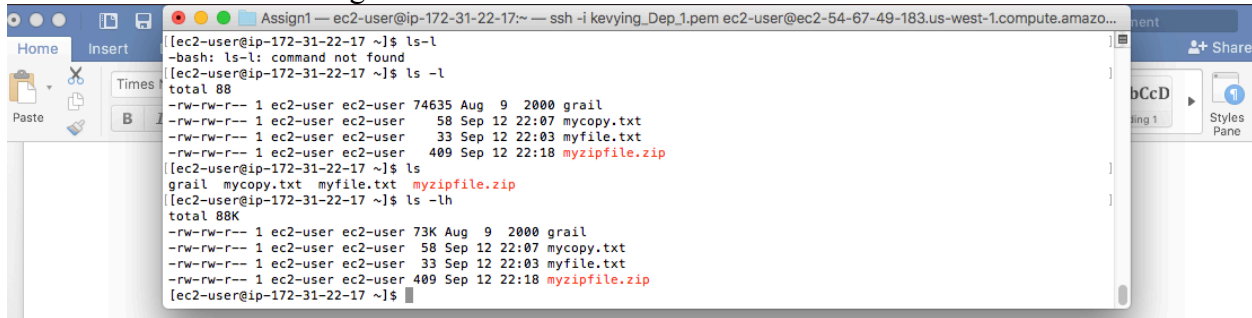
**SUBMIT:** Take a screen shot of the screen after this command (unzip).



A terminal window titled 'Assign1' showing a user navigating to the 'CSC555' directory, creating a zip file, moving it, and then unzipping it. The terminal output is as follows:

```
ec2-user@ip-172-31-22-17 ~]$ ls
ec2-user
ec2-user@ip-172-31-22-17 home]$ ls
ec2-user
ec2-user@ip-172-31-22-17 home]$ cd ec2-user
ec2-user@ip-172-31-22-17 ~]$ ls
CSC555
ec2-user@ip-172-31-22-17 ~]$ cd CSC555
ec2-user@ip-172-31-22-17 CSC555]$ ls
mycopy.txt  myfile.txt  myzipfile.zip
ec2-user@ip-172-31-22-17 CSC555]$ mv myzipfile.zip /home/ec2-user/
ec2-user@ip-172-31-22-17 CSC555]$ ls
mycopy.txt  myfile.txt
ec2-user@ip-172-31-22-17 CSC555]$ cd ..
ec2-user@ip-172-31-22-17 ~]$ cd
ec2-user@ip-172-31-22-17 ~]$ ls
CSC555  myzipfile.zip
ec2-user@ip-172-31-22-17 ~]$ unzip myzipfile.zip
Archive: myzipfile.zip
  extracting: mycopy.txt
  extracting: myfile.txt
ec2-user@ip-172-31-22-17 ~]$
```

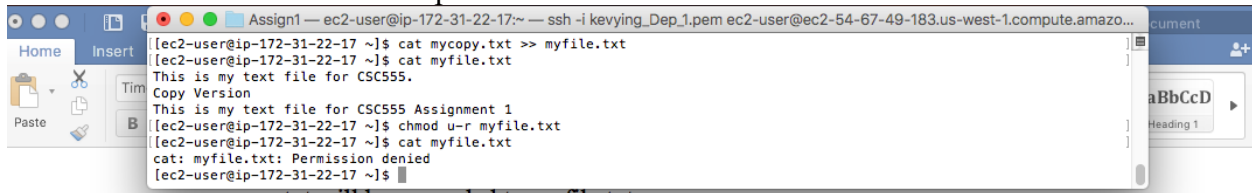
**SUBMIT:** The size of the grail file.



```
[ec2-user@ip-172-31-22-17 ~]$ ls -l
-bash: ls-l: command not found
[ec2-user@ip-172-31-22-17 ~]$ ls -l
total 88
-rw-rw-r-- 1 ec2-user ec2-user 74635 Aug  9 2000 grail
-rw-rw-r-- 1 ec2-user ec2-user  58 Sep 12 22:07 mycopy.txt
-rw-rw-r-- 1 ec2-user ec2-user  33 Sep 12 22:03 myfile.txt
-rw-rw-r-- 1 ec2-user ec2-user 409 Sep 12 22:18 myzipfile.zip
[ec2-user@ip-172-31-22-17 ~]$ ls
grail  mycopy.txt  myfile.txt  myzipfile.zip
[ec2-user@ip-172-31-22-17 ~]$ ls -lh
total 88K
-rw-rw-r-- 1 ec2-user ec2-user 73K Aug  9 2000 grail
-rw-rw-r-- 1 ec2-user ec2-user 58 Sep 12 22:07 mycopy.txt
-rw-rw-r-- 1 ec2-user ec2-user 33 Sep 12 22:03 myfile.txt
-rw-rw-r-- 1 ec2-user ec2-user 409 Sep 12 22:18 myzipfile.zip
[ec2-user@ip-172-31-22-17 ~]$
```

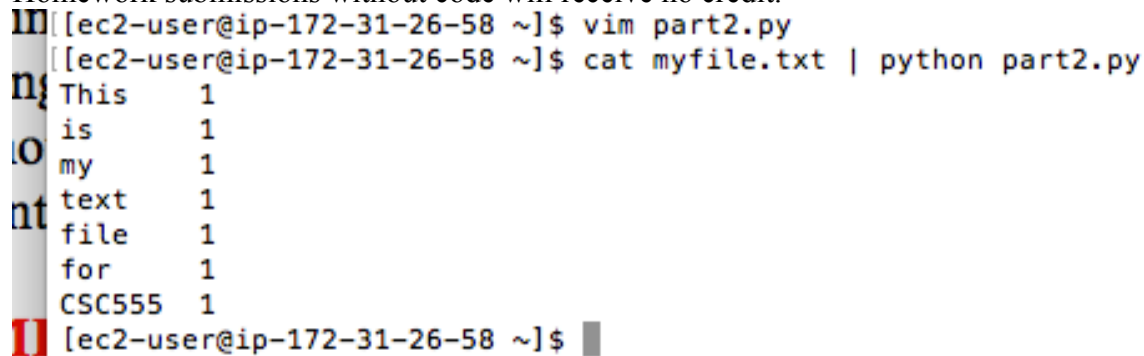
Based on the above, the size of the grail file is about 73K bytes

**SUBMIT:** The screenshot of the permission denied error



```
[ec2-user@ip-172-31-22-17 ~]$ cat mycopy.txt >> myfile.txt
[ec2-user@ip-172-31-22-17 ~]$ cat myfile.txt
This is my text file for CSC555.
Copy Version
This is my text file for CSC555 Assignment 1
[ec2-user@ip-172-31-22-17 ~]$ chmod u-r myfile.txt
[ec2-user@ip-172-31-22-17 ~]$ cat myfile.txt
cat: myfile.txt: Permission denied
[ec2-user@ip-172-31-22-17 ~]$
```

**SUBMIT:** The screen output from running your python code and a copy of your python code.  
Homework submissions without code will receive no credit.



```
[ec2-user@ip-172-31-26-58 ~]$ vim part2.py
[ec2-user@ip-172-31-26-58 ~]$ cat myfile.txt | python part2.py
This      1
is        1
my        1
text      1
file      1
for       1
CSC555    1
[ec2-user@ip-172-31-26-58 ~]$
```

Code:

```
import sys

for line in sys.stdin:
    line=line.strip()
    words=line.split(' ')

    for word in words:
        print '%s\t%s' %(word,1)
```

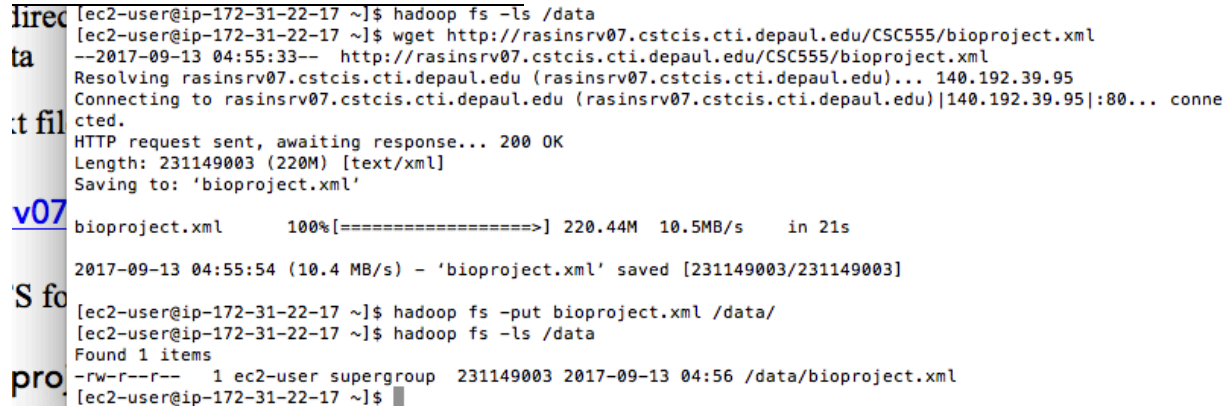
Part 3:

Copy the file to HDFS for processing

`hadoop fs -put bioproject.xml /data/`

(you can optimally verify that the file was uploaded to HDFS by `hadoop fs -ls /data`)

**Submit a screenshot of this command**



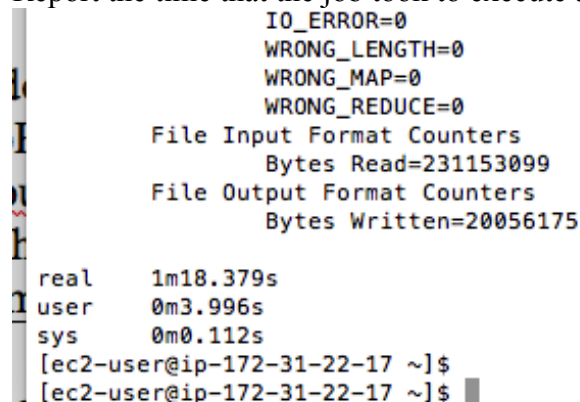
```
[ec2-user@ip-172-31-22-17 ~]$ hadoop fs -ls /data
[ec2-user@ip-172-31-22-17 ~]$ wget http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/bioproject.xml
--2017-09-13 04:55:33-- http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/bioproject.xml
Resolving rasinsrv07.cstcis.cti.depaul.edu (rasinsrv07.cstcis.cti.depaul.edu)... 140.192.39.95
Connecting to rasinsrv07.cstcis.cti.depaul.edu (rasinsrv07.cstcis.cti.depaul.edu)|140.192.39.95|:80... conne
cted.
HTTP request sent, awaiting response... 200 OK
Length: 231149003 (220M) [text/xml]
Saving to: 'bioproject.xml'

bioproject.xml      100%[=====>] 220.44M  10.5MB/s   in 21s

2017-09-13 04:55:54 (10.4 MB/s) - 'bioproject.xml' saved [231149003/231149003]

[ec2-user@ip-172-31-22-17 ~]$ hadoop fs -put bioproject.xml /data/
[ec2-user@ip-172-31-22-17 ~]$ hadoop fs -ls /data
Found 1 items
-rw-r--r--  1 ec2-user supergroup 231149003 2017-09-13 04:56 /data/bioproject.xml
[ec2-user@ip-172-31-22-17 ~]$
```

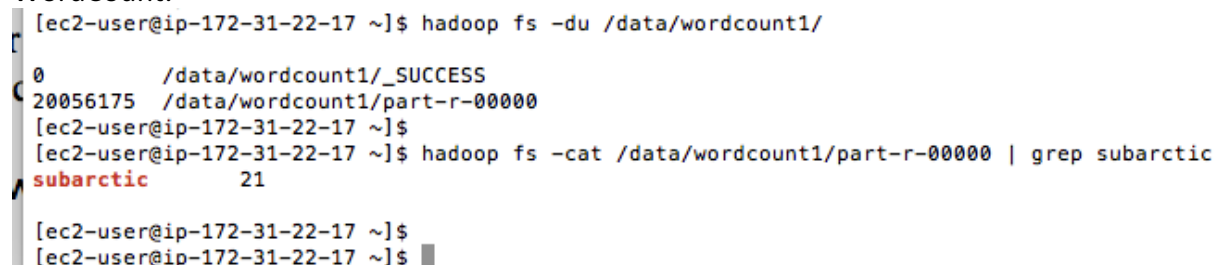
Report the time that the job took to execute as screenshot



```
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=231153099
File Output Format Counters
  Bytes Written=20056175

real    1m18.379s
user    0m3.996s
sys     0m0.112s
[ec2-user@ip-172-31-22-17 ~]$
[ec2-user@ip-172-31-22-17 ~]$
```

**WordCount:**



```
[ec2-user@ip-172-31-22-17 ~]$ hadoop fs -du /data/wordcount1/
0          /data/wordcount1/_SUCCESS
20056175   /data/wordcount1/part-r-00000
[ec2-user@ip-172-31-22-17 ~]$
[ec2-user@ip-172-31-22-17 ~]$ hadoop fs -cat /data/wordcount1/part-r-00000 | grep subarctic
subarctic      21
[ec2-user@ip-172-31-22-17 ~]$
[ec2-user@ip-172-31-22-17 ~]$
```

ent of part-r-00000 file and then uses pipe | operator to filter it