# <u>Alex Teboul - CSC 555 Assignment 1</u>

**CSC 555: Mining Big Data**

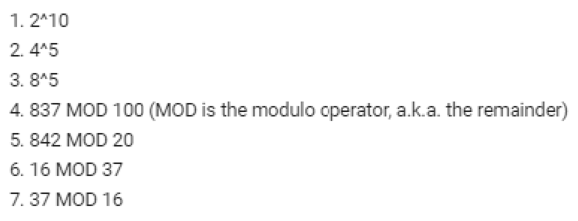
Assignment 1 (due Tuesday, January 21st)

Suggested reading: *Mining of Massive Datasets*: Chapter 1, Chapter 2 (sections 2.1, 2.1 only).
*Hadoop: The Definitive Guide*: Appendix A (available on D2L)
Supplemental document **UsingAmazonAWS.doc**. My Code from this Homework @ Link:
https://colab.research.google.com/drive/1uSBbqYDuBm1MD55nkJTW9tANDo63grc1

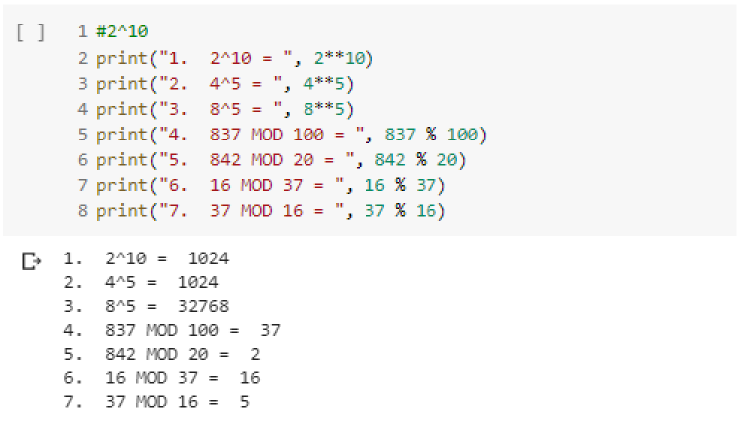
## Part 1

a) **Compute (you can use any tool to compute answers in this part – but if you do not know to perform this computation, please talk to me about prerequisites):**

$2^{10}$
$4^5$
$8^5$
837 MOD 100 (MOD is the modulo operator, a.k.a. the remainder)
842 MOD 20
16 MOD 37
37 MOD 16



1. 2^10
2. 4^5
3. 8^5
4. 837 MOD 100 (MOD is the modulo operator, a.k.a. the remainder)
5. 842 MOD 20
6. 16 MOD 37
7. 37 MOD 16



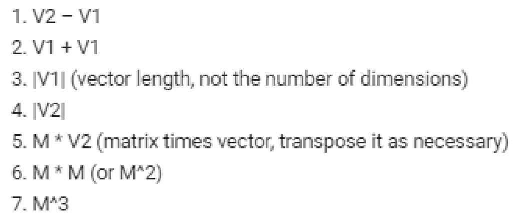
```
#2^10
print("1.  2^10 = ", 2**10)
print("2.  4^5 = ", 4**5)
print("3.  8^5 = ", 8**5)
print("4.  837 MOD 100 = ", 837 % 100)
print("5.  842 MOD 20 = ", 842 % 20)
print("6.  16 MOD 37 = ", 16 % 37)
print("7.  37 MOD 16 = ", 37 % 16)
```

```
1.  2^10 =  1024
2.  4^5 =  1024
3.  8^5 =  32768
4.  837 MOD 100 =  37
5.  842 MOD 20 =  2
6.  16 MOD 37 =  16
7.  37 MOD 16 =  5
```

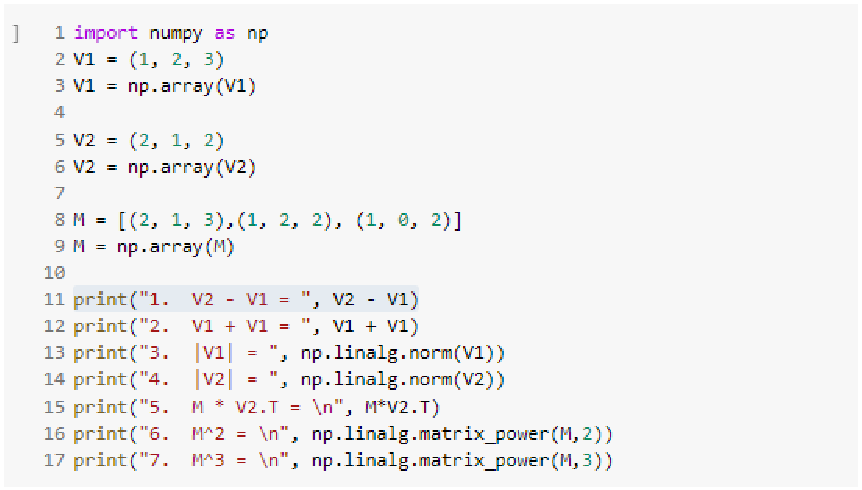
b) **Given vectors V1 = (1, 2, 3) and V2 = (2, 1, 2) and a 3x3 matrix M = [(2, 1, 3), (1, 2, 2), (1, 0, 2)], compute:**

V2 – V1
V1 + V1
|V1| (vector length, not the number of dimensions)

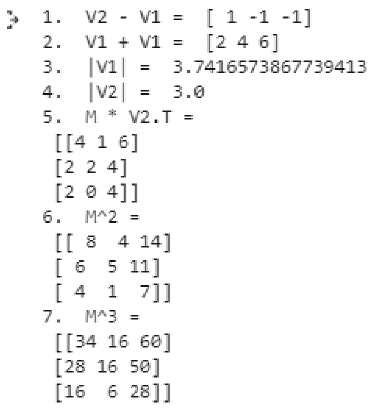
|V2|
M * V2 (matrix times vector, transpose it as necessary)
M * M (or $M^2$)
$M^3$



1. V2 – V1
2. V1 + V1
3. |V1| (vector length, not the number of dimensions)
4. |V2|
5. M * V2 (matrix times vector, transpose it as necessary)
6. M * M (or M^2)
7. M^3



```python
import numpy as np
V1 = (1, 2, 3)
V1 = np.array(V1)

V2 = (2, 1, 2)
V2 = np.array(V2)

M = [(2, 1, 3),(1, 2, 2), (1, 0, 2)]
M = np.array(M)

print("1.  V2 - V1 = ", V2 - V1)
print("2.  V1 + V1 = ", V1 + V1)
print("3.  |V1| = ", np.linalg.norm(V1))
print("4.  |V2| = ", np.linalg.norm(V2))
print("5.  M * V2.T = \n", M*V2.T)
print("6.  M^2 = \n", np.linalg.matrix_power(M,2))
print("7.  M^3 = \n", np.linalg.matrix_power(M,3))
```

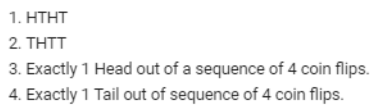


```
1.  V2 - V1 =  [ 1 -1 -1]
2.  V1 + V1 =  [2 4 6]
3.  |V1|  =  3.7416573867739413
4.  |V2|  =  3.0
5.  M * V2.T =
 [[4 1 6]
 [2 2 4]
 [2 0 4]]
6.  M^2 =
 [[ 8  4 14]
 [ 6  5 11]
 [ 4  1  7]]
7.  M^3 =
 [[34 16 60]
 [28 16 50]
 [16  6 28]]
```

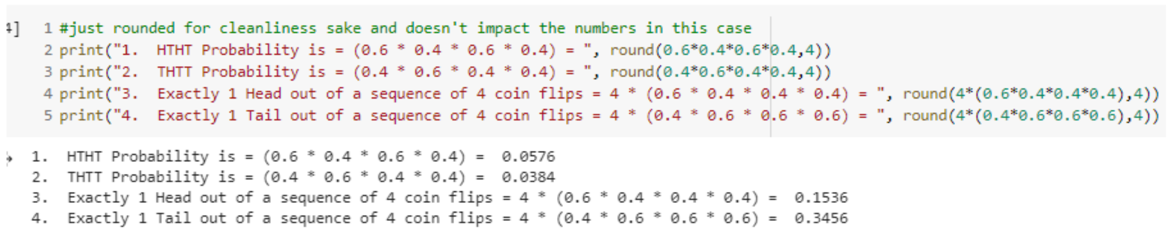
- I assume that by vector length you mean the same as magnitude.

**c) Suppose we are flipping a coin with Head (H) and Tail (T) sides. The coin is <u>not</u> balanced with 0.6 probability of H coming up (and 0.4 of T). Compute the probabilities of getting:**

HTHT
THTT
Exactly 1 Head out of a sequence of 4 coin flips.
Exactly 1 Tail out of sequence of 4 coin flips.



1. HTHT
2. THTT
3. Exactly 1 Head out of a sequence of 4 coin flips.
4. Exactly 1 Tail out of sequence of 4 coin flips.



```
#just rounded for cleanliness sake and doesn't impact the numbers in this case
print("1.  HTHT Probability is = (0.6 * 0.4 * 0.6 * 0.4) = ", round(0.6*0.4*0.6*0.4,4))
print("2.  THTT Probability is = (0.4 * 0.6 * 0.4 * 0.4) = ", round(0.4*0.6*0.4*0.4,4))
print("3.  Exactly 1 Head out of a sequence of 4 coin flips = 4 * (0.6 * 0.4 * 0.4 * 0.4) = ", round(4*(0.6*0.4*0.4*0.4),4))
print("4.  Exactly 1 Tail out of a sequence of 4 coin flips = 4 * (0.4 * 0.6 * 0.6 * 0.6) = ", round(4*(0.4*0.6*0.6*0.6),4))
```

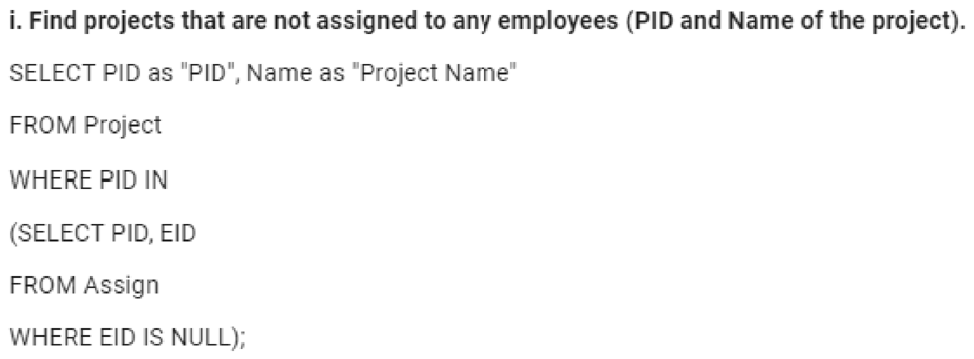
```
1.  HTHT Probability is = (0.6 * 0.4 * 0.6 * 0.4) =  0.0576
2.  THTT Probability is = (0.4 * 0.6 * 0.4 * 0.4) =  0.0384
3.  Exactly 1 Head out of a sequence of 4 coin flips = 4 * (0.6 * 0.4 * 0.4 * 0.4) =  0.1536
4.  Exactly 1 Tail out of a sequence of 4 coin flips = 4 * (0.4 * 0.6 * 0.6 * 0.6) =  0.3456
```

**Answers:** 0.0576, 0.0384, 0.1536, 0.3456

d) **Consider a database schema consisting of two tables, Employee (<u>ID</u>, Name, Address), Project (<u>PID</u>, Name, Deadline), Assign(<u>EID</u>, <u>PID</u>, Date). Assign.EID is a foreign key referencing employee's ID and Assign.PID is a foreign key referencing the project.**

Write SQL queries for:

i. **Find projects that are not assigned to any employees (PID and Name of the project).**



**i. Find projects that are not assigned to any employees (PID and Name of the project).**

SELECT PID as "PID", Name as "Project Name"

FROM Project

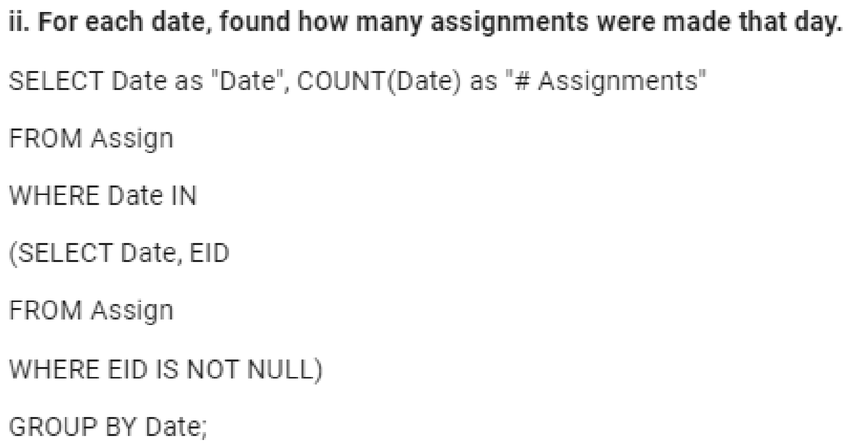
WHERE PID IN

(SELECT PID, EID

FROM Assign

WHERE EID IS NULL);

ii. **For each date, found how many assignments were made that day.**



**ii. For each date, found how many assignments were made that day.**

SELECT Date as "Date", COUNT(Date) as "# Assignments"

FROM Assign

WHERE Date IN

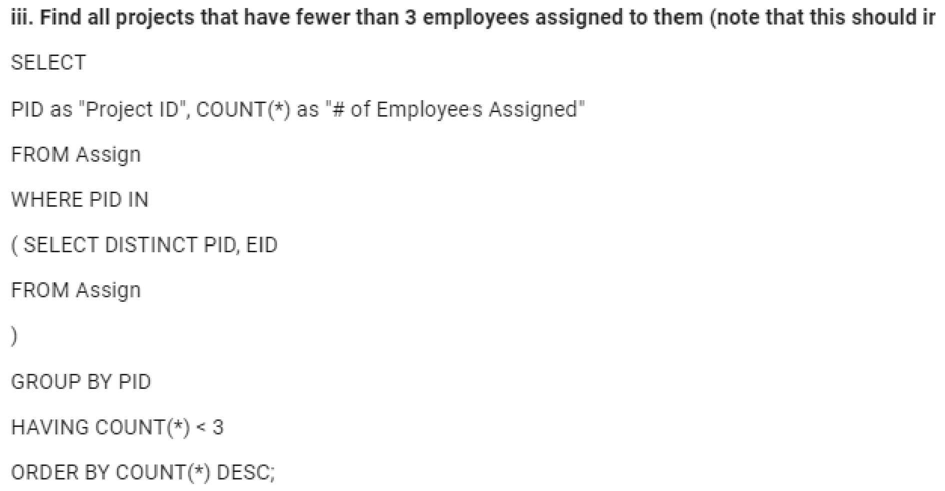
(SELECT Date, EID

FROM Assign

WHERE EID IS NOT NULL)

GROUP BY Date;

iii. **Find all projects that have fewer than 3 employees assigned to them (note that this should include 2, 1 and 0 in order to be correct)**



iii. Find all projects that have fewer than 3 employees assigned to them (note that this should ir

```
SELECT
PID as "Project ID", COUNT(*) as "# of Employees Assigned"
FROM Assign
WHERE PID IN
( SELECT DISTINCT PID, EID
FROM Assign
)
GROUP BY PID
HAVING COUNT(*) < 3
ORDER BY COUNT(*) DESC;
```

*Strange query because it seems like you can get multiple assignments to the same project if it's entered on different Assign.Dates. ^I think this should work despite that though.

e) **Mining of Massive Datasets, Exercise 1.3.3**
**<u>Justify your answer</u>.**

**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 mod 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?

- In order for the hash function h to send approximately equal numbers of hash-keys to each of the buckets, they need to be chosen randomly from the population of possible hash-keys.
- Also, "The generalization of Example 1.4 is that when hash-keys are integers, chosing B so it has any common factor with all (or even most of) the possible hashkeys will result in nonrandom distribution into buckets."
- We know that h is x % 15 and that the population of possible hash keys must be > the number of buckets.
- But B=15 seems like a bad choice because it is not a prime number...
- We know that the hash keys are positive integers.
- Following the logic presented in this chapter, for h(x) = x mod B, where B is the number of buckets, B=15 in this exercise.
- Granted this is my first time learning about hashing, but it seems like c=1 would work to get 0 - 14 buckets filled, but c=2 would not get values into the even buckets.
- By this logic c could also be 1, 16, 31, 46, 61... (1, B+1, 2B+1, 3B+1, 4B+1...) for the same effect.
- From the textbook and other sources online, this doesn't seem like a good option do just do x mod 15 as the only hashing function used. At least use a prime B seems to be better practice.

f) **Describe how you would implement a MapReduce job consisting of Map and Reduce description. You do not have to write code or even pseudo-code. Just describe, in your own words, what the Map and Reduce tasks are going to do. Map task reads the input file and produces (key, value) pairs. Reduce task takes a list of (key, value) pairs for each key and combines all values for each key.**
**Please remember that Map operates on individual blocks and Reduce on individual keys with a set of values. Thus, for Mapper you need to state what your code does given a block of data (i.e., for each block, not for the whole file) and for Reduce you need to state what your reducer does for each key (without being able to see other keys).**
**For a data file that contains the following columns: (ID, First, Last, Grade)**

i. For each first name, find the GPA (grade point average) of each student, i.e., **SELECT First, AVG(Grade) FROM Student GROUP BY First**.

- **Map:** The mapper creates the pair of First and Grade, such that the key is the student's first name and their grades are the values.
- **Reduce:** The reducer looks at all the grades of that student, and gets the average = sum(grades)/count(grades).

ii. For each full student name, find the best grade, i.e., **SELECT First, Last, MAX(Grade) FROM Student GROUP BY First, Last**.

- **Map:** The mapper creates joins First and Last, then creates the pair of FirstLast and Grade, such that the key is the student's First&Last name and their grades are the values.
- **Reduce:** The reducer looks at all the grades of that student, and gets the top or MAX grade from the list of grades for that student. Not sure if this task involves sorting to find the max.

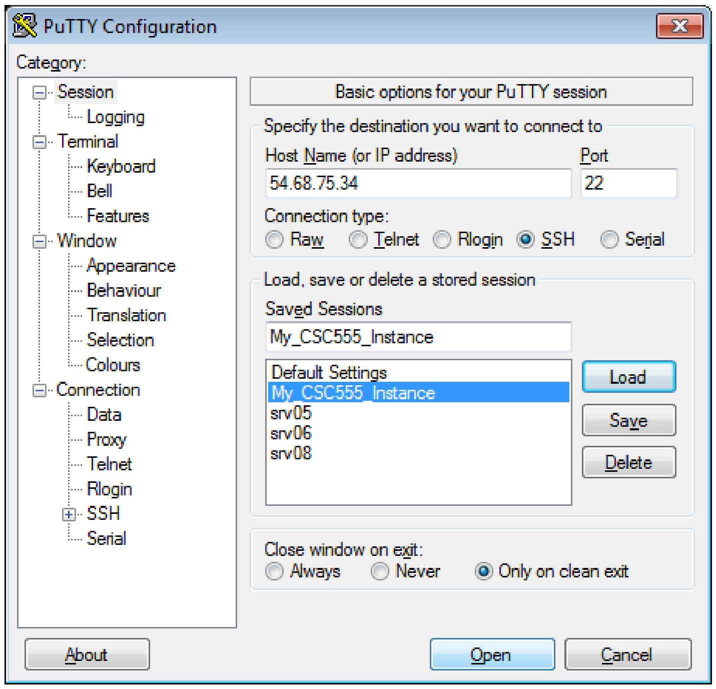
## Part 2: Linux Intro

This part of the assignment will serve as an introduction to Linux. Make sure you go through the steps below and submit screenshots where requested – submit the entire screenshot of a command terminal.

Use at least a t2.small instance or Hadoop may not run properly with insufficient memory.
All Linux commands are in Berlin Sans FB. Do not type the "$" symbol. The "$" represents the prompt "[ec2-user@ip-xxx-xx-xx-xxx ~] $ " in your particular Linux instance.

**0. Login to your Amazon EC2 Instance** (NOTE: instructions on how to create a new instance and log in to it are provided in a separate file, UsingAmazonAWS.doc)

Connect to your instance through Putty.



Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): 54.68.75.34

Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

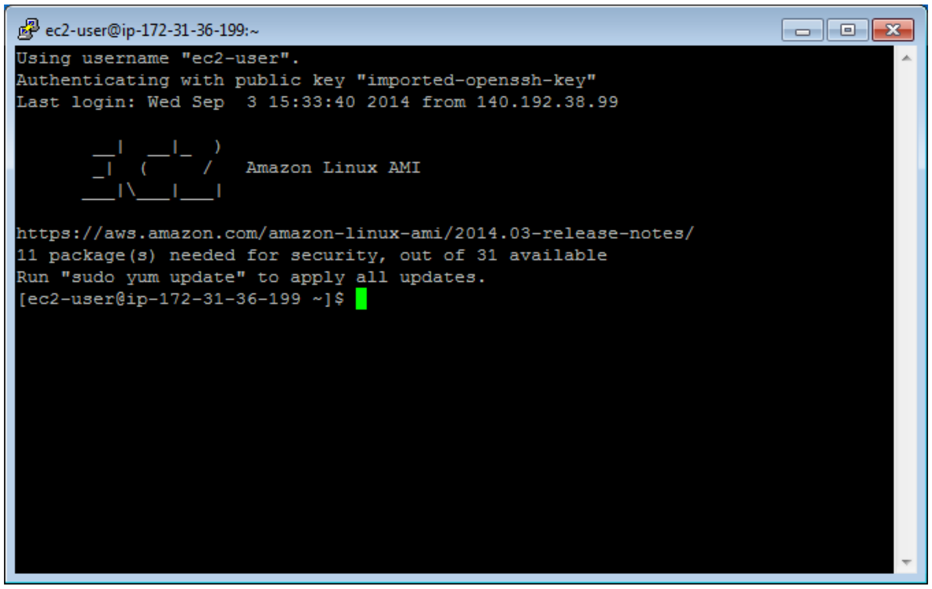
Load, save or delete a stored session

Saved Sessions: My_CSC555_Instance

Default Settings
My_CSC555_Instance
srv05
srv06
srv08

Close window on exit: Always Never [✓ Only on clean exit]

Your instance should look like as the following image:



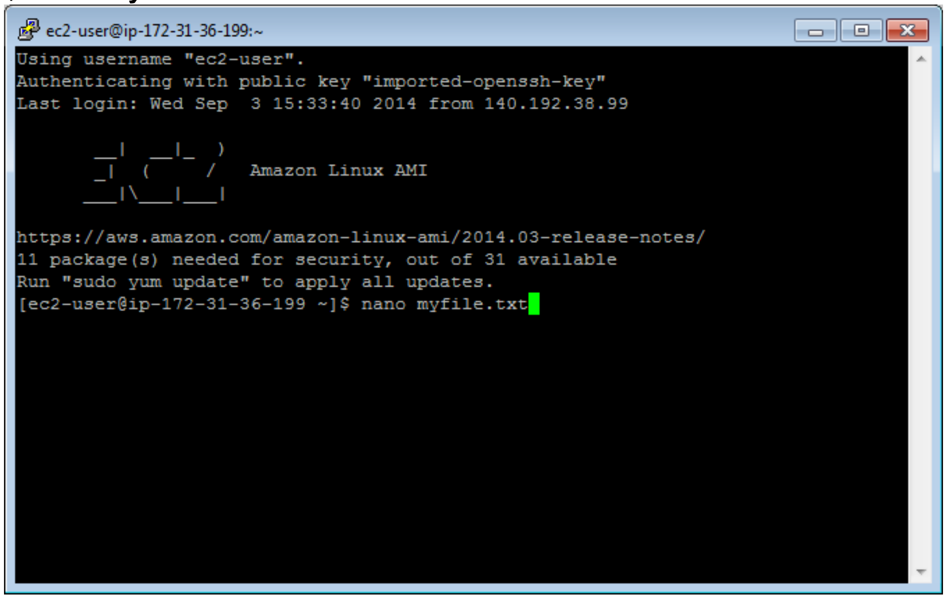
```
Using username "ec2-user".
Authenticating with public key "imported-openssh-key"
Last login: Wed Sep  3 15:33:40 2014 from 140.192.38.99

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2014.03-release-notes/
11 package(s) needed for security, out of 31 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-36-199 ~]$ 
```

1. **Create a text file.**
   Instructions for 3 different text editors are provided below. You only need to choose one editor that you prefer. **nano** is a more basic text editor, and is much easier to start. **vim** and **emacs** are more advanced and rely on keyboard shortcuts quite a bit and thus have a steeper learning curve.

• **Tip**: To paste into Linux terminal you can use right-click. To copy from the Linux terminal, you only need to highlight the text that you want to copy with your mouse. Also please remember that Linux is case-sensitive, which means **Nano** and **nano** are not equivalent.

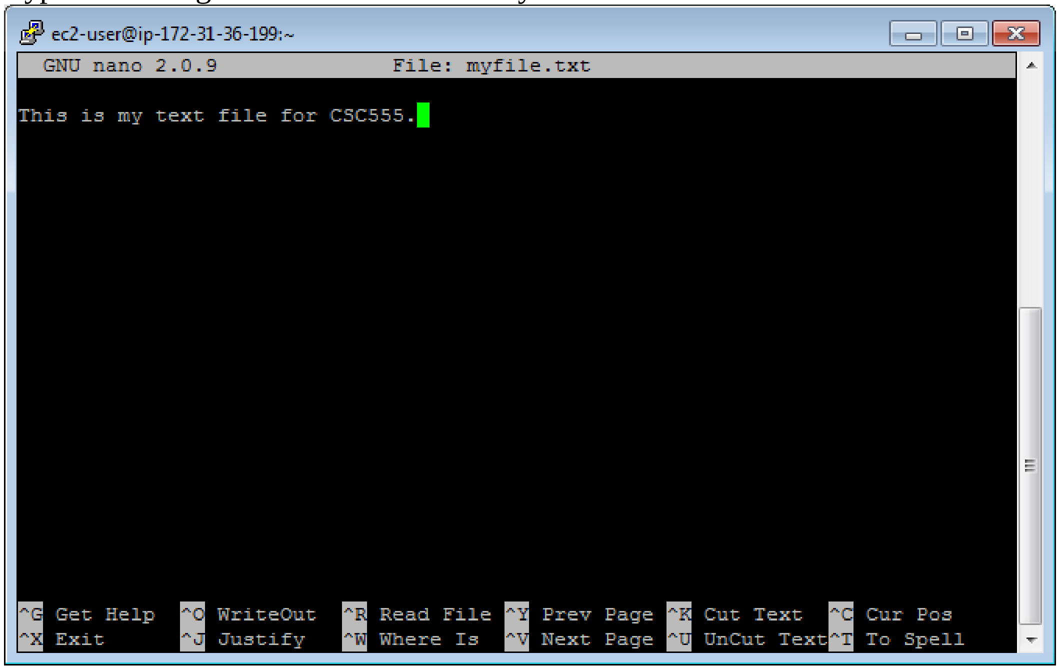
Nano Instructions(Option 1):

**$ nano myfile.txt**



```
Using username "ec2-user".
Authenticating with public key "imported-openssh-key"
Last login: Wed Sep  3 15:33:40 2014 from 140.192.38.99

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2014.03-release-notes/
11 package(s) needed for security, out of 31 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-36-199 ~]$ nano myfile.txt
```

Type something into the file: "This is my text file for CSC555."



```
  GNU nano 2.0.9               File: myfile.txt

This is my text file for CSC555.

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

Save changes: Ctrl-o and hit Enter.
Exit: Ctrl-x

Emacs Instructions(Option 2):

You will need to install emacs.
**$ sudo yum install emacs**
Type “y” when asked if this is OK to install.

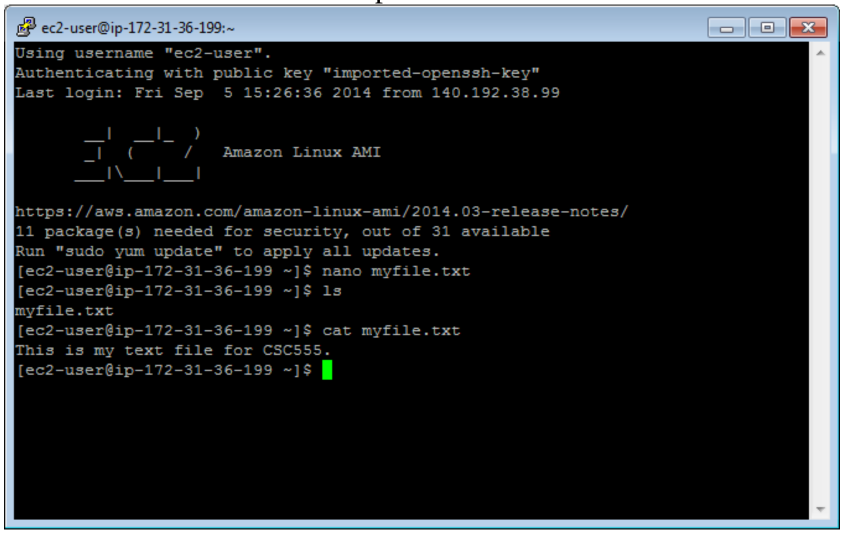
**$ emacs myfile.txt**
Type something into the file: “This is my text file for CSC555.”
Save changes: Ctrl-x, Ctrl-s
Exit: Ctrl-x Ctrl-z

Vim Instructions(Option 3):
• NOTE: When **vim** opens, you are in *command* mode. Any key you enter will be bound to a command instead of inserted into the file. To enter *insert* mode press the key “i”. To save a file or exit you will need to hit Esc to get back into *command* mode.

**$ vim myfile.txt**
Type “i” to enter *insert* mode
Type something into the file: “This is my text file for CSC555.”
Save changes: hit Esc to enter *command* mode then type “:w”
Exit: (still in *command* mode) type “:x”

Confirm your file has been saved by listing the files in the working directory.
**$ ls**
You should see your file.
Display the contents of the file on the screen.
**$ cat myfile.txt**

Your file contents should be printed to the terminal.



```
Using username "ec2-user".
Authenticating with public key "imported-openssh-key"
Last login: Fri Sep  5 15:26:36 2014 from 140.192.38.99

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2014.03-release-notes/
11 package(s) needed for security, out of 31 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-36-199 ~]$ nano myfile.txt
[ec2-user@ip-172-31-36-199 ~]$ ls
myfile.txt
[ec2-user@ip-172-31-36-199 ~]$ cat myfile.txt
This is my text file for CSC555.
[ec2-user@ip-172-31-36-199 ~]$
```

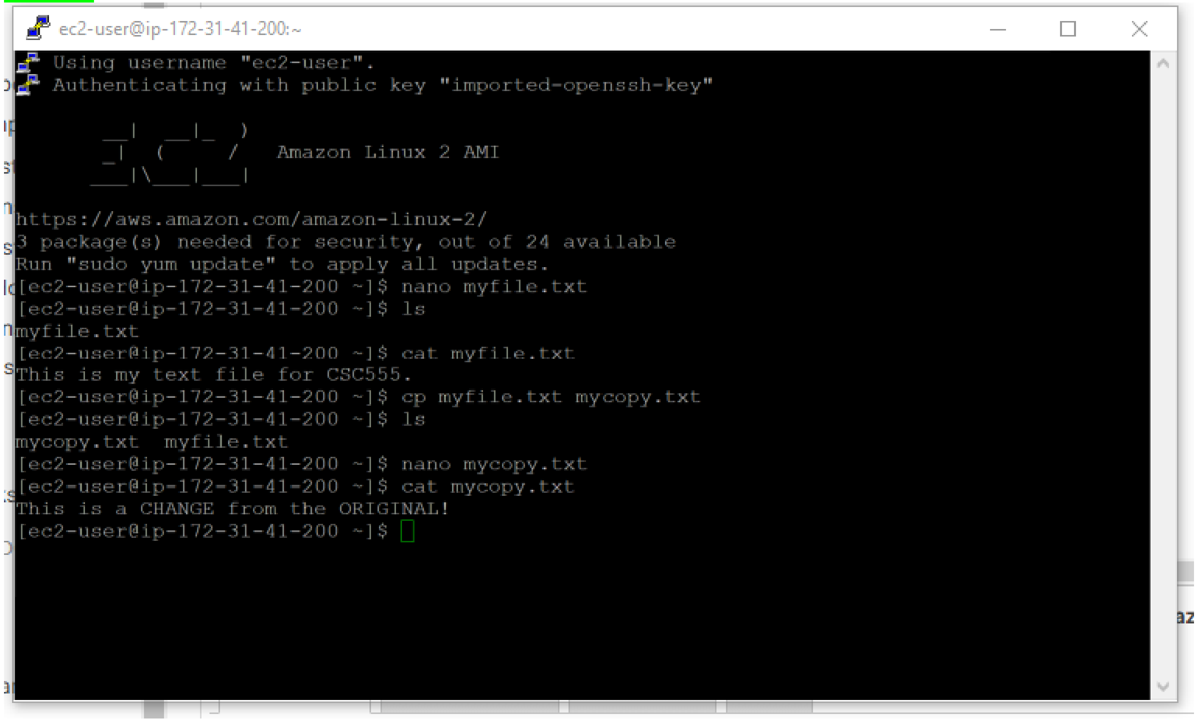
• **Tip**: Linux will fill in partially typed commands if you hit Tab.
  **$cat myfi**
  Hit Tab and “myfi” should be completed to “myfile.txt”. If there are multiple completion options, hit Tab twice and a list of all possible completions will be printed. This also

applies to commands themselves, i.e. you can type in **ca** and see all possible commands that begin with ca.

**2. Copy your file.**

Make a copy.
**$ cp myfile.txt mycopy.txt**
Confirm this file has been created by listing the files in the working directory.
Edit this file so it contains different text than the original file using the text editor instructions, and confirm your changes by displaying the contents of the file on the screen.

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



```
 Using username "ec2-user".
 Authenticating with public key "imported-openssh-key"

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
3 package(s) needed for security, out of 24 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-41-200 ~]$ nano myfile.txt
[ec2-user@ip-172-31-41-200 ~]$ ls
myfile.txt
[ec2-user@ip-172-31-41-200 ~]$ cat myfile.txt
This is my text file for CSC555.
[ec2-user@ip-172-31-41-200 ~]$ cp myfile.txt mycopy.txt
[ec2-user@ip-172-31-41-200 ~]$ ls
mycopy.txt  myfile.txt
[ec2-user@ip-172-31-41-200 ~]$ nano mycopy.txt
[ec2-user@ip-172-31-41-200 ~]$ cat mycopy.txt
This is a CHANGE from the ORIGINAL!
[ec2-user@ip-172-31-41-200 ~]$
```

**3. Delete a file**

Make a copy to delete.
**$ cp myfile.txt filetodelete.txt**
**$ ls**

Remove the file.
**$ rm filetodelete.txt**
**$ ls**

**4. Create a directory to put your files.**

Make a directory.

**$mkdir CSC555**

Change the current directory to your new directory.
**$cd CSC555**

Print your current working directory
**$pwd**

**5. Move your files to your new directory.**

Return to your home directory.
**$cd**
OR
**$cd ..**
OR
**$ cd /home/ec2-user/**

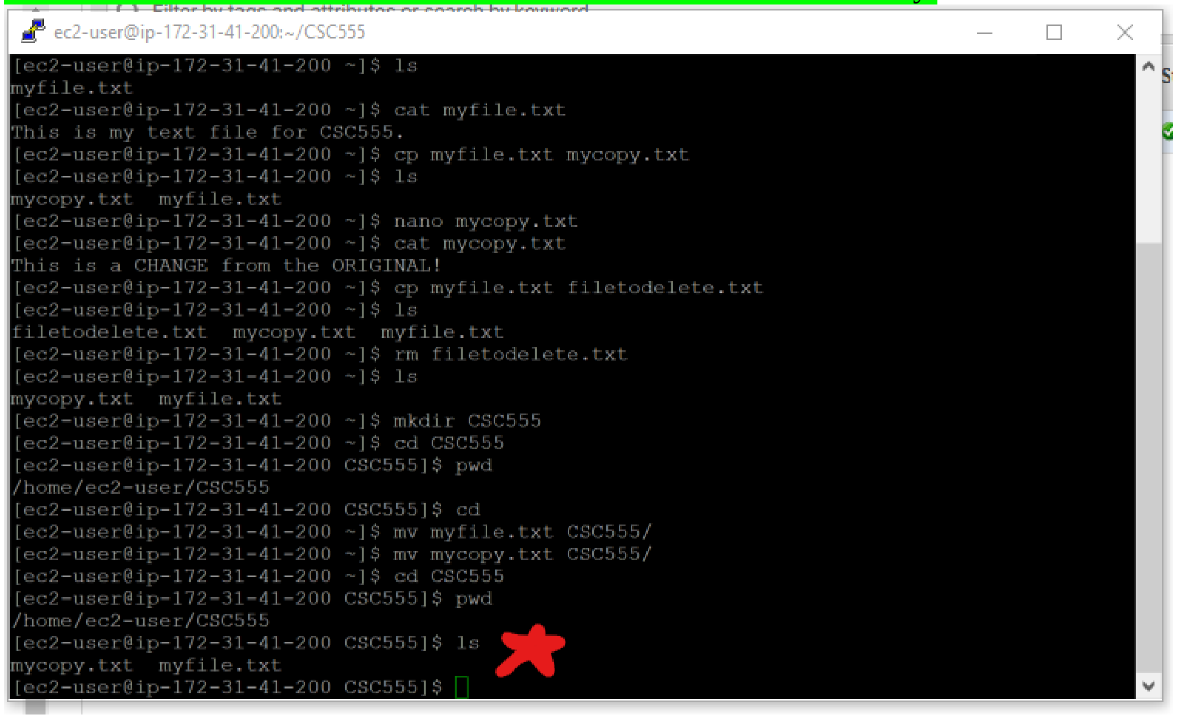
• NOTE: **cd** will always take you to your home directory. **cd ..** will move you up one directory level (to the parent). Your home directory is "/home/[user name]", /home/ec2-user in our case

Move your files to your new directory.

**$ mv myfile.txt CSC555/**
**$ mv mycopy.txt CSC555/**

Change the current directory to CSC555 and list the files in this directory.

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



```
[ec2-user@ip-172-31-41-200 ~]$ ls
myfile.txt
[ec2-user@ip-172-31-41-200 ~]$ cat myfile.txt
This is my text file for CSC555.
[ec2-user@ip-172-31-41-200 ~]$ cp myfile.txt mycopy.txt
[ec2-user@ip-172-31-41-200 ~]$ ls
mycopy.txt  myfile.txt
[ec2-user@ip-172-31-41-200 ~]$ nano mycopy.txt
[ec2-user@ip-172-31-41-200 ~]$ cat mycopy.txt
This is a CHANGE from the ORIGINAL!
[ec2-user@ip-172-31-41-200 ~]$ cp myfile.txt filetodelete.txt
[ec2-user@ip-172-31-41-200 ~]$ ls
filetodelete.txt  mycopy.txt  myfile.txt
[ec2-user@ip-172-31-41-200 ~]$ rm filetodelete.txt
[ec2-user@ip-172-31-41-200 ~]$ ls
mycopy.txt  myfile.txt
[ec2-user@ip-172-31-41-200 ~]$ mkdir CSC555
[ec2-user@ip-172-31-41-200 ~]$ cd CSC555
[ec2-user@ip-172-31-41-200 CSC555]$ pwd
/home/ec2-user/CSC555
[ec2-user@ip-172-31-41-200 CSC555]$ cd
[ec2-user@ip-172-31-41-200 ~]$ mv myfile.txt CSC555/
[ec2-user@ip-172-31-41-200 ~]$ mv mycopy.txt CSC555/
[ec2-user@ip-172-31-41-200 ~]$ cd CSC555
[ec2-user@ip-172-31-41-200 CSC555]$ pwd
/home/ec2-user/CSC555
[ec2-user@ip-172-31-41-200 CSC555]$ ls
mycopy.txt  myfile.txt
[ec2-user@ip-172-31-41-200 CSC555]$
```

## 6. Zip and Unzip your files.

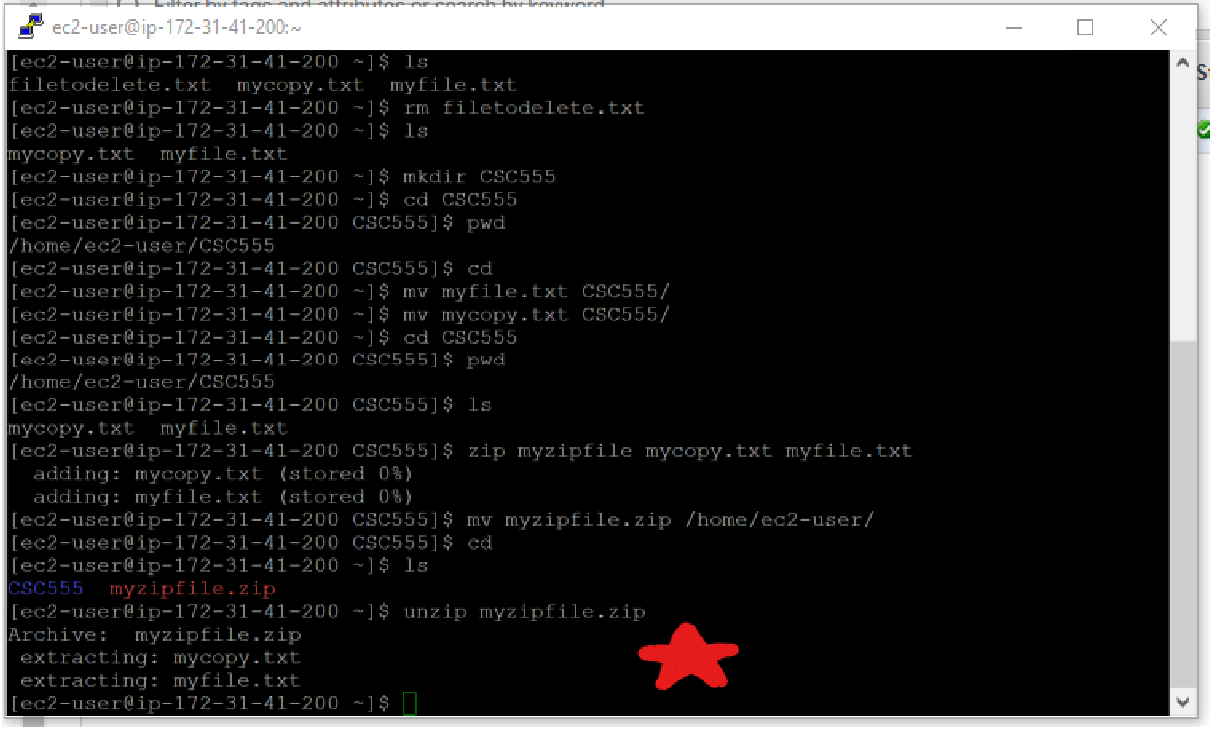
Zip the files.
**$ zip myzipfile mycopy.txt myfile.txt**
OR
**$ zip myzipfile ***

• NOTE: * is the wildcard symbol that matches everything in current directory. If there should be any additional files in the current directory, they will also be placed into the zip archive. Wildcard can also be used to match files selectively. For example **zip myzipfile my*** will zip-up all files beginning with "my" in the current directory.

Move your zip file to your home directory.
**$ mv myzipfile.zip /home/ec2-user/**

Return to your home directory.
Extract the files.
**$ unzip myzipfile.zip**

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



```
[ec2-user@ip-172-31-41-200 ~]$ ls
filetodelete.txt  mycopy.txt  myfile.txt
[ec2-user@ip-172-31-41-200 ~]$ rm filetodelete.txt
[ec2-user@ip-172-31-41-200 ~]$ ls
mycopy.txt  myfile.txt
[ec2-user@ip-172-31-41-200 ~]$ mkdir CSC555
[ec2-user@ip-172-31-41-200 ~]$ cd CSC555
[ec2-user@ip-172-31-41-200 CSC555]$ pwd
/home/ec2-user/CSC555
[ec2-user@ip-172-31-41-200 CSC555]$ cd
[ec2-user@ip-172-31-41-200 ~]$ mv myfile.txt CSC555/
[ec2-user@ip-172-31-41-200 ~]$ mv mycopy.txt CSC555/
[ec2-user@ip-172-31-41-200 ~]$ cd CSC555
[ec2-user@ip-172-31-41-200 CSC555]$ pwd
/home/ec2-user/CSC555
[ec2-user@ip-172-31-41-200 CSC555]$ ls
mycopy.txt  myfile.txt
[ec2-user@ip-172-31-41-200 CSC555]$ zip myzipfile mycopy.txt myfile.txt
  adding: mycopy.txt (stored 0%)
  adding: myfile.txt (stored 0%)
[ec2-user@ip-172-31-41-200 CSC555]$ mv myzipfile.zip /home/ec2-user/
[ec2-user@ip-172-31-41-200 CSC555]$ cd
[ec2-user@ip-172-31-41-200 ~]$ ls
CSC555  myzipfile.zip
[ec2-user@ip-172-31-41-200 ~]$ unzip myzipfile.zip
Archive:  myzipfile.zip
 extracting: mycopy.txt
 extracting: myfile.txt
[ec2-user@ip-172-31-41-200 ~]$
```

## 7. Remove your CSC555 directory.

• **Warning**: Executing "rm -rf" has the potential to delete ALL files in a given directory, including sub-directories ("r" stands for recursive). You should use this command very carefully.

Delete your CSC555 directory.
**$ rm -rf CSC555/**

**8. Download a file from the web.**

Download the script for Monty Python and the Holy Grail.
**$ wget http://www.textfiles.com/media/SCRIPTS/grail**

The file should be saved as “grail” by default.

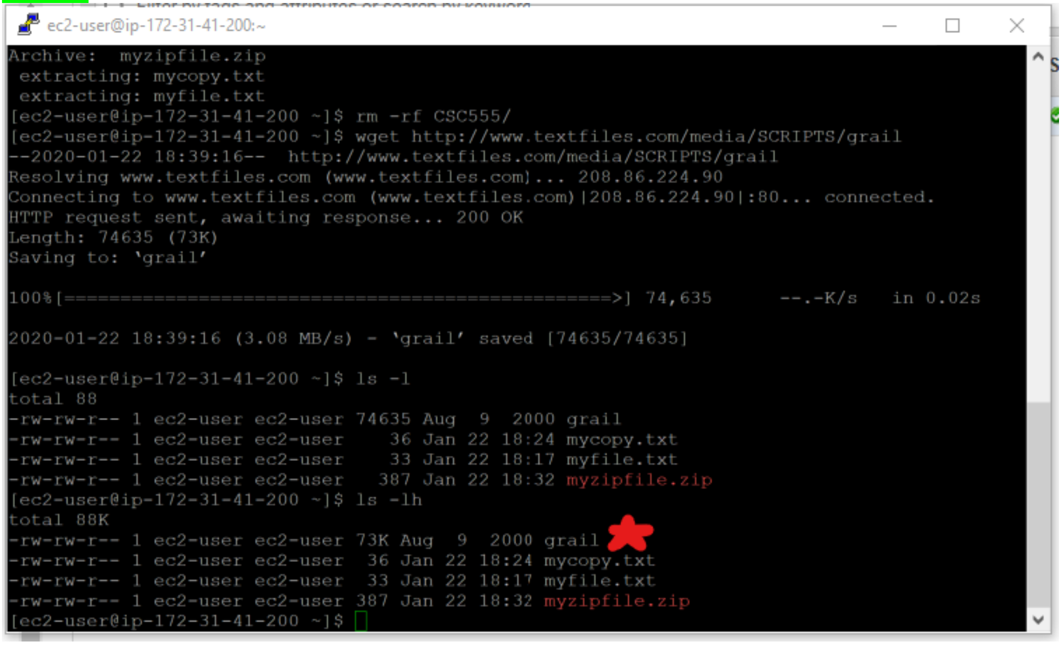
**9. ls formats**

List all contents of the current directory in long list format.
• **Note**: the option following “ls” is the character “l”; not “one”.
**$ ls -l**

The 1st column gives information regarding file permissions (which we will discuss in more detail later). For now, note that the first character of the 10 total will be “-“ for normal files and “d” for directories. The 2nd Column is the number of links to the file. The 3rd and 4th columns are the owner and the group of the file. The 5th column displays the size of the file in bytes. The 6th column is the date and time the file was last modified. The 7th column is the file or directory name.

List all contents of the current directory in long list and human readable formats. “-h” will put large files in more readable units than bytes.
**$ ls -lh**

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


ec2-user@ip-172-31-41-200:~

```
Archive:  myzipfile.zip
 extracting: mycopy.txt
 extracting: myfile.txt
[ec2-user@ip-172-31-41-200 ~]$ rm -rf CSC555/
[ec2-user@ip-172-31-41-200 ~]$ wget http://www.textfiles.com/media/SCRIPTS/grail
--2020-01-22 18:39:16--  http://www.textfiles.com/media/SCRIPTS/grail
Resolving www.textfiles.com (www.textfiles.com)... 208.86.224.90
Connecting to www.textfiles.com (www.textfiles.com)|208.86.224.90|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 74635 (73K)
Saving to: 'grail'

100%[================================================>] 74,635      --.-K/s   in 0.02s

2020-01-22 18:39:16 (3.08 MB/s) - 'grail' saved [74635/74635]

[ec2-user@ip-172-31-41-200 ~]$ 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    36 Jan 22 18:24 mycopy.txt
-rw-rw-r-- 1 ec2-user ec2-user    33 Jan 22 18:17 myfile.txt
-rw-rw-r-- 1 ec2-user ec2-user   387 Jan 22 18:32 myzipfile.zip
[ec2-user@ip-172-31-41-200 ~]$ 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  36 Jan 22 18:24 mycopy.txt
-rw-rw-r-- 1 ec2-user ec2-user  33 Jan 22 18:17 myfile.txt
-rw-rw-r-- 1 ec2-user ec2-user 387 Jan 22 18:32 myzipfile.zip
[ec2-user@ip-172-31-41-200 ~]$
```

## 10. More on viewing files.

If you issue “cat grail”, the contents of grail will be printed. However, this file is too large to fit on the screen.

Show the grail file one page at a time.

```
$ more grail
```

Hit the spacebar to go to the next page. Type “b” to go page up, hit “space” key to go page down. Type “q” to quit.

OR

```
$ less grail
```

*Less* has more options than *more* (“*less* is more and *more* is less”). You can now use the keyboard Arrows and Page Up/Down to scroll. You can type “h” for help, which will display additional options.

View the line numbers in the grail file. The *cat* command has the -n option, which prints line numbers, but you may also want to use *more* to view the file one page at a time. A solution is to *pipe* the output from *cat* to *more*. A *pipe* redirects the output from one program to another program for further processing, and it is represented with “|”.

```
$ cat -n grail | more
```

Redirect the standard output (stdout) of a command.

```
$ cat myfile.txt > redirect1.txt
$ ls -lh > redirect2.txt
```

Append the stdout to a file.

```
$ cat mycopy.txt >> myfile.txt
```

mycopy.txt will be appended to myfile.txt.

• **Note**: “cat mycopy.txt > myfile.txt” will <u>overwrite</u> myfile.txt with the contents output by “cat mycopy.txt”. Thus using >> is crucial if you want to preserve the existing file contents.

## 11. Change access permissions to objects with the *change mode* command.

The following represent roles:
u – user, g – group, o – others, a - all

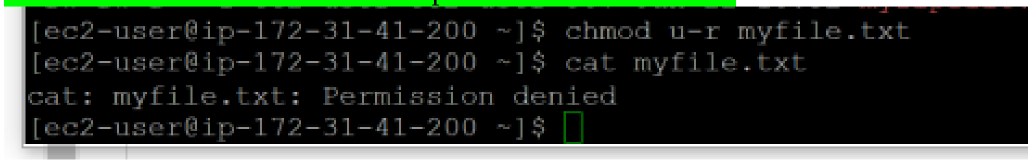
The following represent permissions:
r – read, w – write, x – execute

Remove the read permission for your user on a file.

```
$ chmod u-r myfile.txt
```

Try to read this file. You should receive a “permission denied” message because you are the user who owns the file.

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



```
[ec2-user@ip-172-31-41-200 ~]$ chmod u-r myfile.txt
[ec2-user@ip-172-31-41-200 ~]$ cat myfile.txt
cat: myfile.txt: Permission denied
[ec2-user@ip-172-31-41-200 ~]$
```

Give your user read permission on a file. Use the same file you removed the read permission from.

**$ chmod u+r myfile.txt**

You should now be able to read this file again.

**12. Python examples**

Install Python if it is not available on your machine.

**$ sudo yum install python**

Create a Python file. These instructions will use Emacs as a text editor, but you can still chose the text editor you want.

**$ emacs lucky.py**

(Write a simple Python program)

```
print "*"*22
print "My Lucky Numbers".rjust(20)
print "*"*22

for i in range(10):
    lucky_nbr = (i + 1)*2
    print "My lucky number is %s!" % lucky_nbr
```

Run your Python program.

**$ python lucky.py**

Redirect your output to a file

**$ python lucky.py > lucky.txt**

Pipe the stdout from lucky.py to another Python program that will replace “is” with “was”.

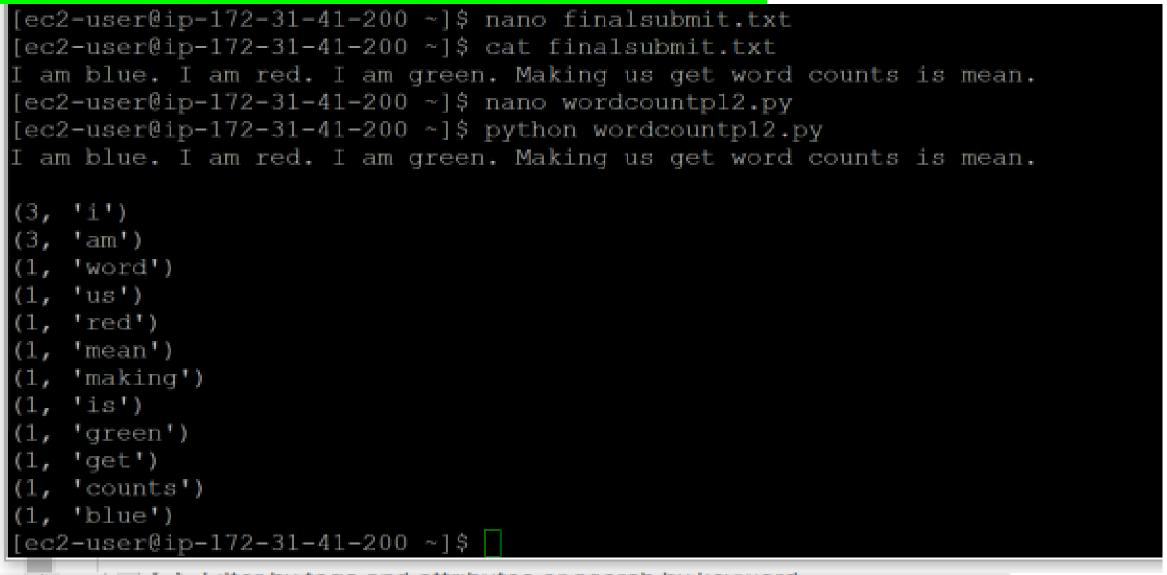
**$ emacs was.py**

```
import sys

for line in sys.stdin:
    print line.replace("is", "was")
```

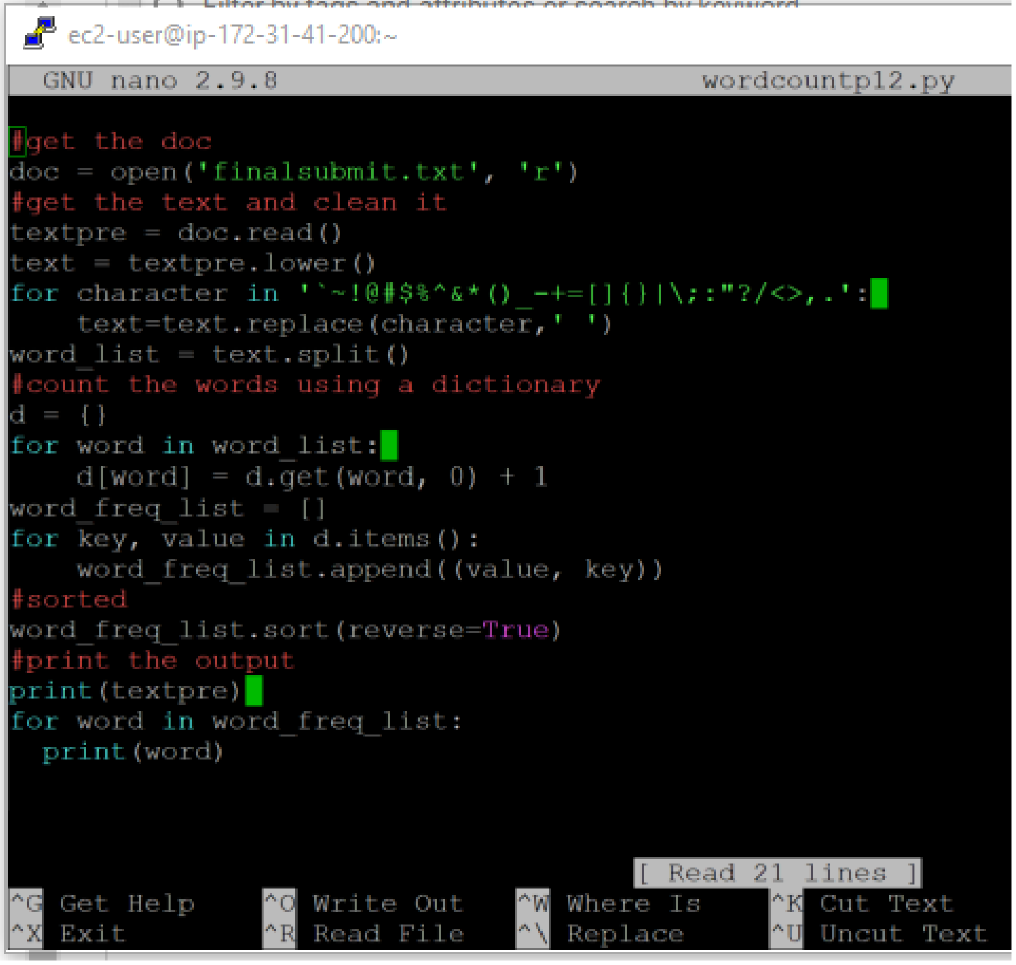
**$ python lucky.py | python was.py**

Write python code to read a text file (you can use myfile.txt) and output word count for each word with the number of times that word occurs in the entire file.

**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-172-31-41-200 ~]$ nano finalsubmit.txt
[ec2-user@ip-172-31-41-200 ~]$ cat finalsubmit.txt
I am blue. I am red. I am green. Making us get word counts is mean.
[ec2-user@ip-172-31-41-200 ~]$ nano wordcountp12.py
[ec2-user@ip-172-31-41-200 ~]$ python wordcountp12.py
I am blue. I am red. I am green. Making us get word counts is mean.

(3, 'i')
(3, 'am')
(1, 'word')
(1, 'us')
(1, 'red')
(1, 'mean')
(1, 'making')
(1, 'is')
(1, 'green')
(1, 'get')
(1, 'counts')
(1, 'blue')
[ec2-user@ip-172-31-41-200 ~]$
```



```
  GNU nano 2.9.8                          wordcountp12.py

#get the doc
doc = open('finalsubmit.txt', 'r')
#get the text and clean it
textpre = doc.read()
text = textpre.lower()
for character in '`~!@#$%^&*()_-+=[]{}|\;:"?/<>,.':
    text=text.replace(character,' ')
word_list = text.split()
#count the words using a dictionary
d = {}
for word in word_list:
    d[word] = d.get(word, 0) + 1
word_freq_list = []
for key, value in d.items():
    word_freq_list.append((value, key))
#sorted
word_freq_list.sort(reverse=True)
#print the output
print(textpre)
for word in word_freq_list:
  print(word)

                                             [ Read 21 lines ]
^G Get Help      ^O Write Out     ^W Where Is      ^K Cut Text
^X Exit          ^R Read File     ^\ Replace       ^U Uncut Text
```

## Part 3: Lab

For this part of the assignment, you will run wordcount on a single-node Hadoop instance. I am going to provide detailed instructions to help you get Hadoop running. The instructions are following Hadoop: The Definitive Guide instructions presented in Appendix A: Installing Apache Hadoop.

You can download 2.6.4 from here. You can copy-paste these commands (right-click in PuTTy to paste, but please watch out for error messages and run commands one by one)

Install ant to list java processes

```
sudo yum install ant
```

(wget command stands for "web get" and lets you download files to your instance from a URL link)

```
wget http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/hadoop-2.6.4.tar.gz
```

(unpack the archive)

```
tar xzf hadoop-2.6.4.tar.gz
```

Modify the conf/hadoop-env.sh to add to it the JAVA_HOME configuration
You can open it by running (using nano or your favorite editor instead of nano).

```
nano hadoop-2.6.4/etc/hadoop/hadoop-env.sh
```

Note that the # comments out the line, so you would comment out the original JAVA_HOME line replacing it by the new one as below.

NOTE: you would need to determine the correct Java configuration line by executing the following (underlined) command

```
[ec2-user@ip-172-31-16-63 ~]$ readlink -f $(which java)
```

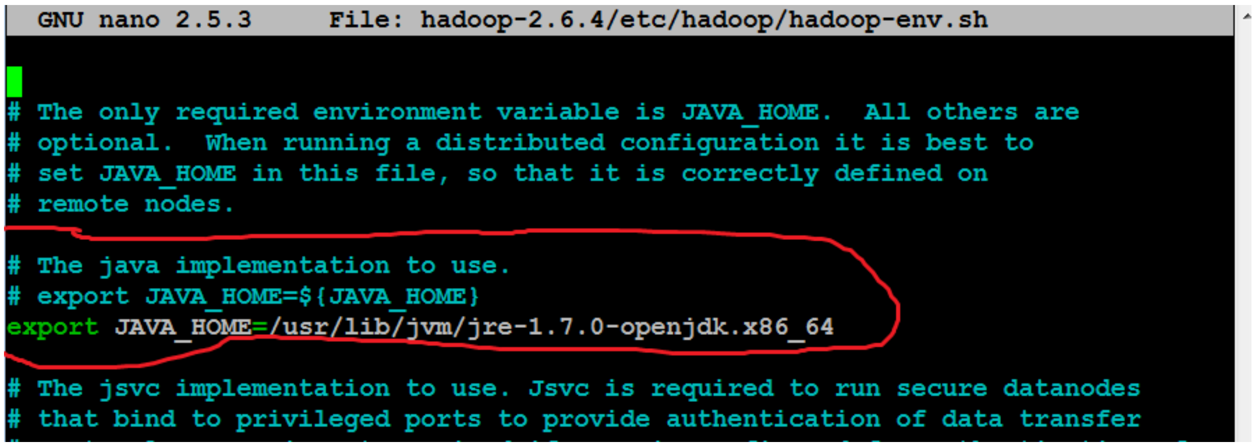
which will output:

```
/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.191.b12-0.amzn2.x86_64/jre/bin/java
```

In my case, Java home is at (remove the bin/java from the output above):

**/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.191.b12-0.amzn2.x86_64/jre/**

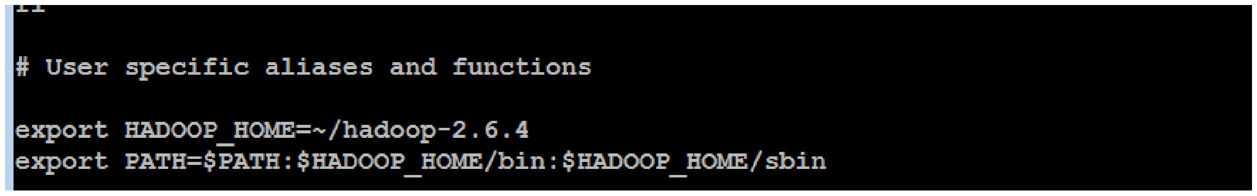
/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.222.b10-0.amzn2.0.1.x86_64/jre/bin/java - mine



```
  GNU nano 2.5.3      File: hadoop-2.6.4/etc/hadoop/hadoop-env.sh


# The only required environment variable is JAVA_HOME.  All others are
# optional.  When running a distributed configuration it is best to
# set JAVA_HOME in this file, so that it is correctly defined on
# remote nodes.

# The java implementation to use.
# export JAVA_HOME=${JAVA_HOME}
export JAVA_HOME=/usr/lib/jvm/jre-1.7.0-openjdk.x86_64

# The jsvc implementation to use. Jsvc is required to run secure datanodes
# that bind to privileged ports to provide authentication of data transfer
```

modify the .bashrc file to add these two lines:
export HADOOP_HOME=~/hadoop-2.6.4
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin

.bashrc file contains environment settings to be configured automatically on each login.
You can open the .bashrc file by running
**nano ~/.bashrc**



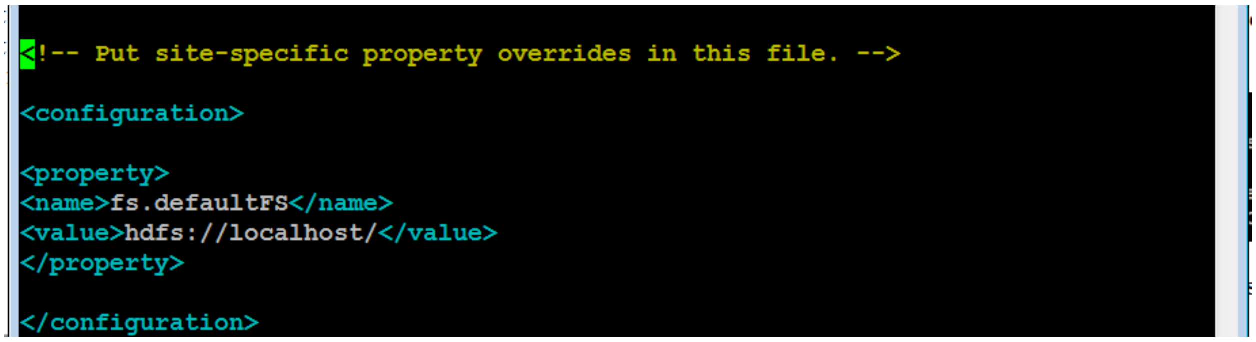
```
# User specific aliases and functions

export HADOOP_HOME=~/hadoop-2.6.4
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
```

To immediately refresh the settings (that will be automatic on next login), run
**source ~/.bashrc**

Next, follow the instructions for Pseudodistributed Mode for all 4 files.

(to edit the first config file)
**nano hadoop-2.6.4/etc/hadoop/core-site.xml**

Make sure you paste the settings between the <configuration> and </configuration> tags, like in the screenshot below. NOTE: The screenshot below is only one of the 4 files, all files are different. The contents of each file are described in the **Appendix A** in the Hadoop book, the relevant appendix is also included with the homework assignment. I am also including a .txt file (HadoopConfigurationText) so that it is easier to copy-paste.



```
<!-- Put site-specific property overrides in this file. -->

<configuration>

<property>
<name>fs.defaultFS</name>
<value>hdfs://localhost/</value>
</property>

</configuration>
```

**nano hadoop-2.6.4/etc/hadoop/hdfs-site.xml**
(mapred-site.xml file is not there, run the following single line command to create it by copying from template. Then you can edit it as other files.)

```
cp hadoop-2.6.4/etc/hadoop/mapred-site.xml.template  hadoop-2.6.4/etc/hadoop/mapred-site.xml
nano hadoop-2.6.4/etc/hadoop/mapred-site.xml
nano hadoop-2.6.4/etc/hadoop/yarn-site.xml
```

To enable passwordless ssh access (we will discuss SSH and public/private keys in class), run these commands:

```
ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

test by running (and confirming yes to a one-time warning)

```
ssh localhost
exit
```

Format HDFS (i.e., first time initialize)

```
hdfs namenode -format
```

Start HDFS, Hadoop and history server (answer a 1-time yes if you asked about host authenticity)

```
start-dfs.sh
start-yarn.sh
mr-jobhistory-daemon.sh start historyserver
```

Verify if everything is running:

```
jps
```

(NameNode and DataNode are responsible for HDFS management; NodeManager and ResourceManager are serving the function similar to JobTracker and TaskTracker. We will discuss function of all of those on Thursday.)

Create a destination directory
hadoop fs -mkdir /data

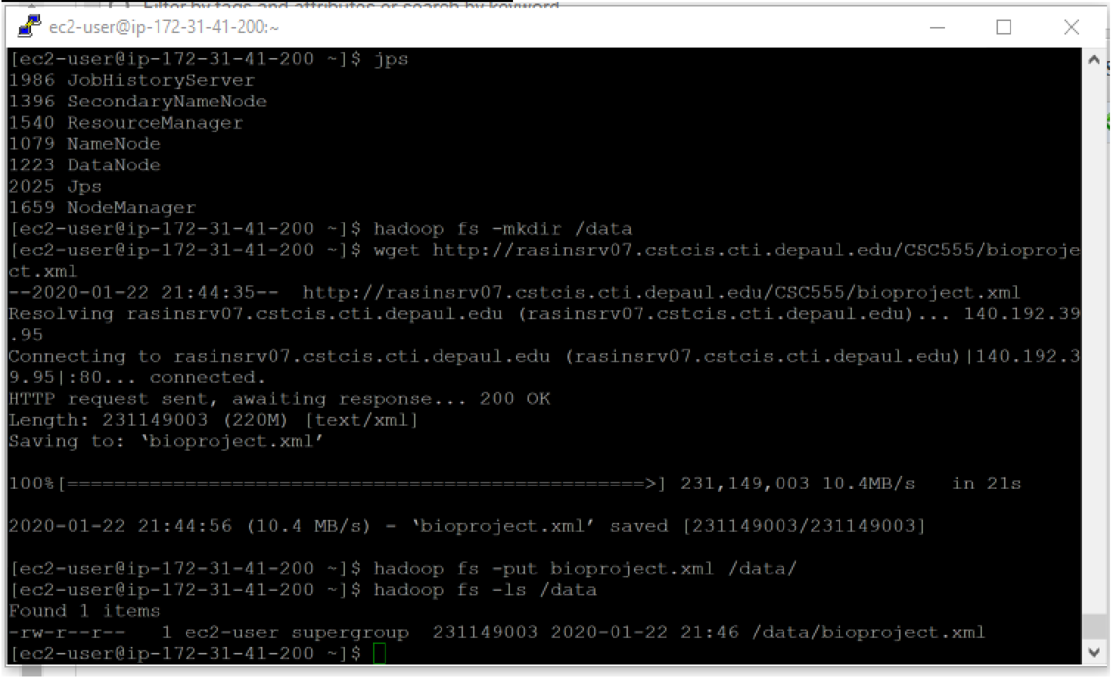
Download a large text file using

**wget http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/bioproject.xml**

Copy the file to HDFS for processing

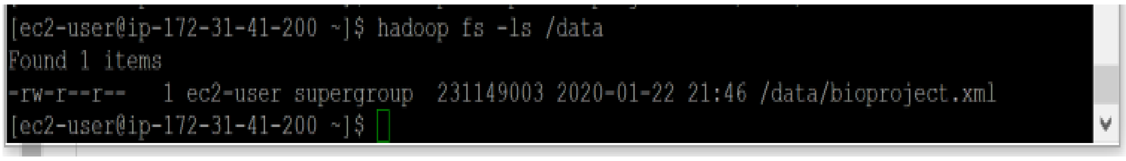
**hadoop fs -put bioproject.xml /data/**

(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-172-31-41-200 ~]$ jps
1986 JobHistoryServer
1396 SecondaryNameNode
1540 ResourceManager
1079 NameNode
1223 DataNode
2025 Jps
1659 NodeManager
[ec2-user@ip-172-31-41-200 ~]$ hadoop fs -mkdir /data
[ec2-user@ip-172-31-41-200 ~]$ wget http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/bioproje
ct.xml
--2020-01-22 21:44:35--  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)|140.192.3
9.95|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 231149003 (220M) [text/xml]
Saving to: 'bioproject.xml'

100%[===============================================>] 231,149,003 10.4MB/s   in 21s

2020-01-22 21:44:56 (10.4 MB/s) - 'bioproject.xml' saved [231149003/231149003]

[ec2-user@ip-172-31-41-200 ~]$ hadoop fs -put bioproject.xml /data/
[ec2-user@ip-172-31-41-200 ~]$ hadoop fs -ls /data
Found 1 items
-rw-r--r--   1 ec2-user supergroup  231149003 2020-01-22 21:46 /data/bioproject.xml
[ec2-user@ip-172-31-41-200 ~]$
```

File is in.

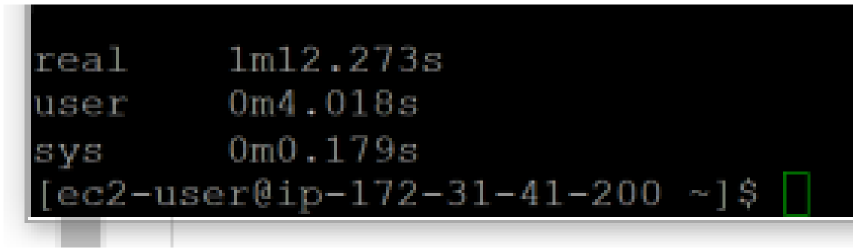


```
[ec2-user@ip-172-31-41-200 ~]$ hadoop fs -ls /data
Found 1 items
-rw-r--r--   1 ec2-user supergroup  231149003 2020-01-22 21:46 /data/bioproject.xml
[ec2-user@ip-172-31-41-200 ~]$
```

Run word count on the downloaded text file, using the time command to determine the total runtime of the MapReduce job.   You can use the following (single-line!) command. This invokes the wordcount example built into the example jar file, supplying /data/bioproject.xml as the input and /data/wordcount1 as the output directory. Please remember this is one command, if you do not paste it as a single line, it will not work.

**time hadoop jar hadoop-2.6.4/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.4.jar  wordcount /data/bioproject.xml /data/wordcount1**

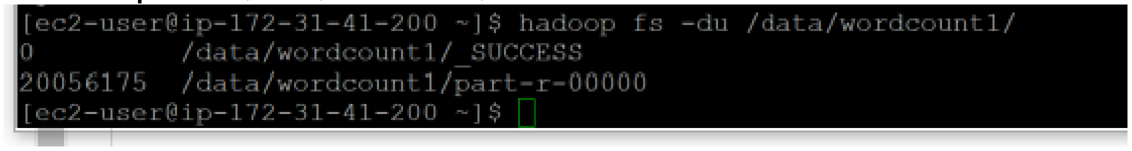
Report the time that the job took to execute as screenshot



```
real    1m12.273s
user    0m4.018s
sys     0m0.179s
[ec2-user@ip-172-31-41-200 ~]$
```

(this reports the size of a particular file or directory in HDFS. The output file will be named part-r-00000)

**hadoop fs -du /data/wordcount1/**

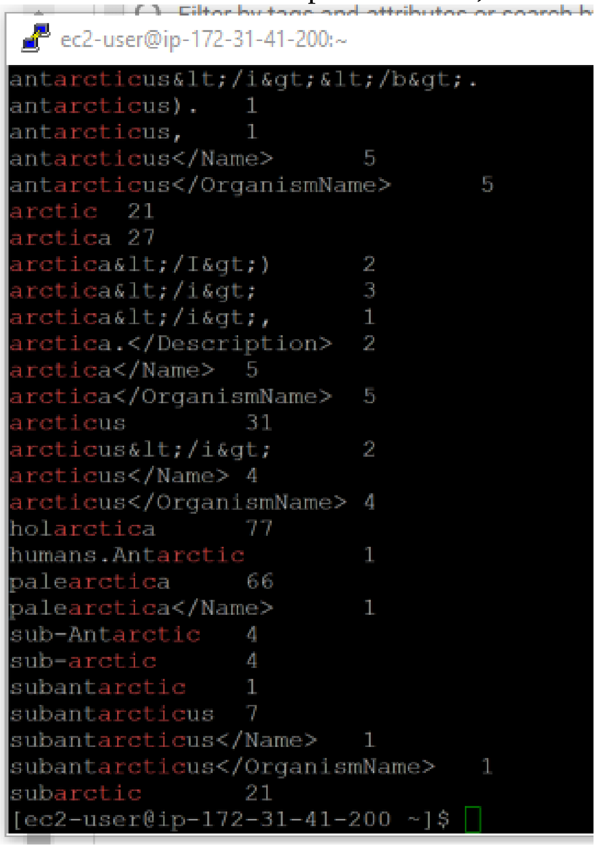


```
[ec2-user@ip-172-31-41-200 ~]$ hadoop fs -du /data/wordcount1/
0         /data/wordcount1/_SUCCESS
20056175  /data/wordcount1/part-r-00000
[ec2-user@ip-172-31-41-200 ~]$
```

(Just like in Linux, the cat HDFS command will dump the output of the entire file and grep command will filter the output to all lines that matches this particular word). To determine the count of occurrences of “arctic”, run the following command:

**hadoop fs -cat /data/wordcount1/part-r-00000 | grep arctic**

It outputs the entire content of part-r-00000 file and then uses pipe | operator to filter it through grep (filter) command. If you remove the pipe and grep, you will get the entire word count content dumped to screen, similar to cat command.



ec2-user@ip-172-31-41-200:~

```
antarcticus&lt;/i&gt;&lt;/b&gt;.
antarcticus).   1
antarcticus,    1
antarcticus</Name>      5
antarcticus</OrganismName>      5
arctic  21
arctica 27
arctica&lt;/I&gt;)       2
arctica&lt;/i&gt;        3
arctica&lt;/i&gt;,       1
arctica.</Description>  2
arctica</Name>  5
arctica</OrganismName>  5
arcticus        31
arcticus&lt;/i&gt;       2
arcticus</Name> 4
arcticus</OrganismName> 4
holarctica      77
humans.Antarctic        1
palearctica     66
palearctica</Name>      1
sub-Antarctic   4
sub-arctic      4
subantarctic    1
subantarcticus  7
subantarcticus</Name>   1
subantarcticus</OrganismName>   1
subarctic       21
[ec2-user@ip-172-31-41-200 ~]$
```

Congratulations, you just finished running wordcount using Hadoop.

<u>Submit a single document containing your written answers.  Be sure that this document contains your name and “CSC 555 Assignment 1” at the top.</u>