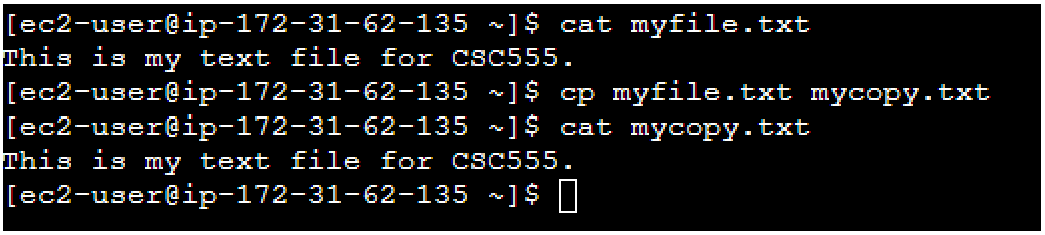
CSC 555: HW1

Name: Nachiketh Reddy

ID: 2117731

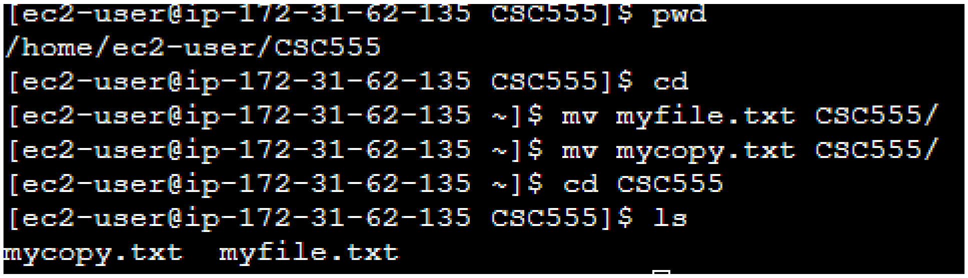
1. **Read the email from AWS Educate Follow the instructions to start your lab, and run an instance of AWS EC2 machine. To create an instance follow instructions from UsingAmazonAWS1 posted on D2L. After you have created an instance, open a terminal and run the Linux Intro Tutorial. Submit screenshots as described in the tutorial.**

1A



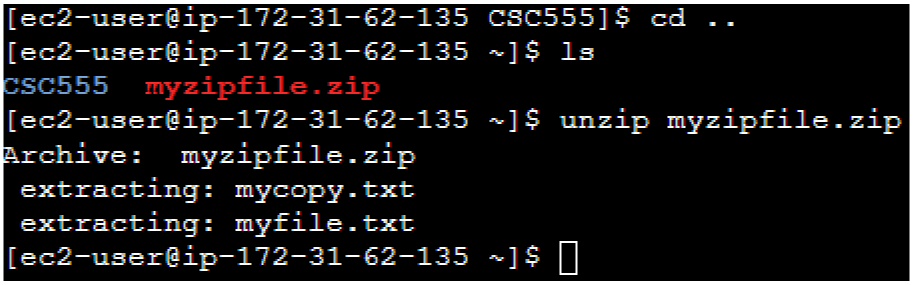
```
[ec2-user@ip-172-31-62-135 ~]$ cat myfile.txt
This is my text file for CSC555.
[ec2-user@ip-172-31-62-135 ~]$ cp myfile.txt mycopy.txt
[ec2-user@ip-172-31-62-135 ~]$ cat mycopy.txt
This is my text file for CSC555.
[ec2-user@ip-172-31-62-135 ~]$
```

1B



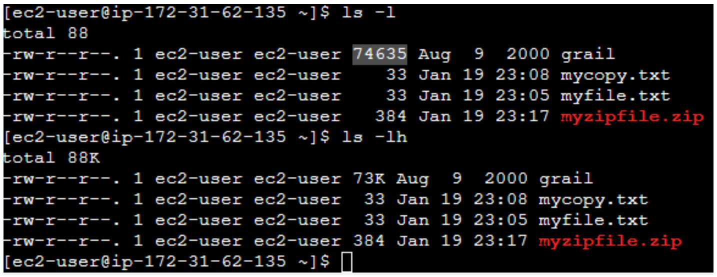
```
[ec2-user@ip-172-31-62-135 CSC555]$ pwd
/home/ec2-user/CSC555
[ec2-user@ip-172-31-62-135 CSC555]$ cd
[ec2-user@ip-172-31-62-135 ~]$ mv myfile.txt CSC555/
[ec2-user@ip-172-31-62-135 ~]$ mv mycopy.txt CSC555/
[ec2-user@ip-172-31-62-135 ~]$ cd CSC555
[ec2-user@ip-172-31-62-135 CSC555]$ ls
mycopy.txt  myfile.txt
```

1C



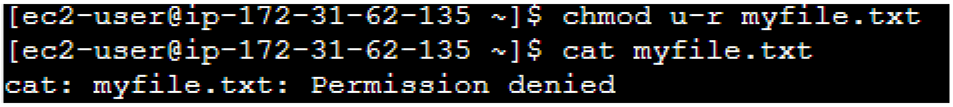
```
[ec2-user@ip-172-31-62-135 CSC555]$ cd ..
[ec2-user@ip-172-31-62-135 ~]$ ls
CSC555  myzipfile.zip
[ec2-user@ip-172-31-62-135 ~]$ unzip myzipfile.zip
Archive:  myzipfile.zip
 extracting: mycopy.txt
 extracting: myfile.txt
[ec2-user@ip-172-31-62-135 ~]$
```

1D



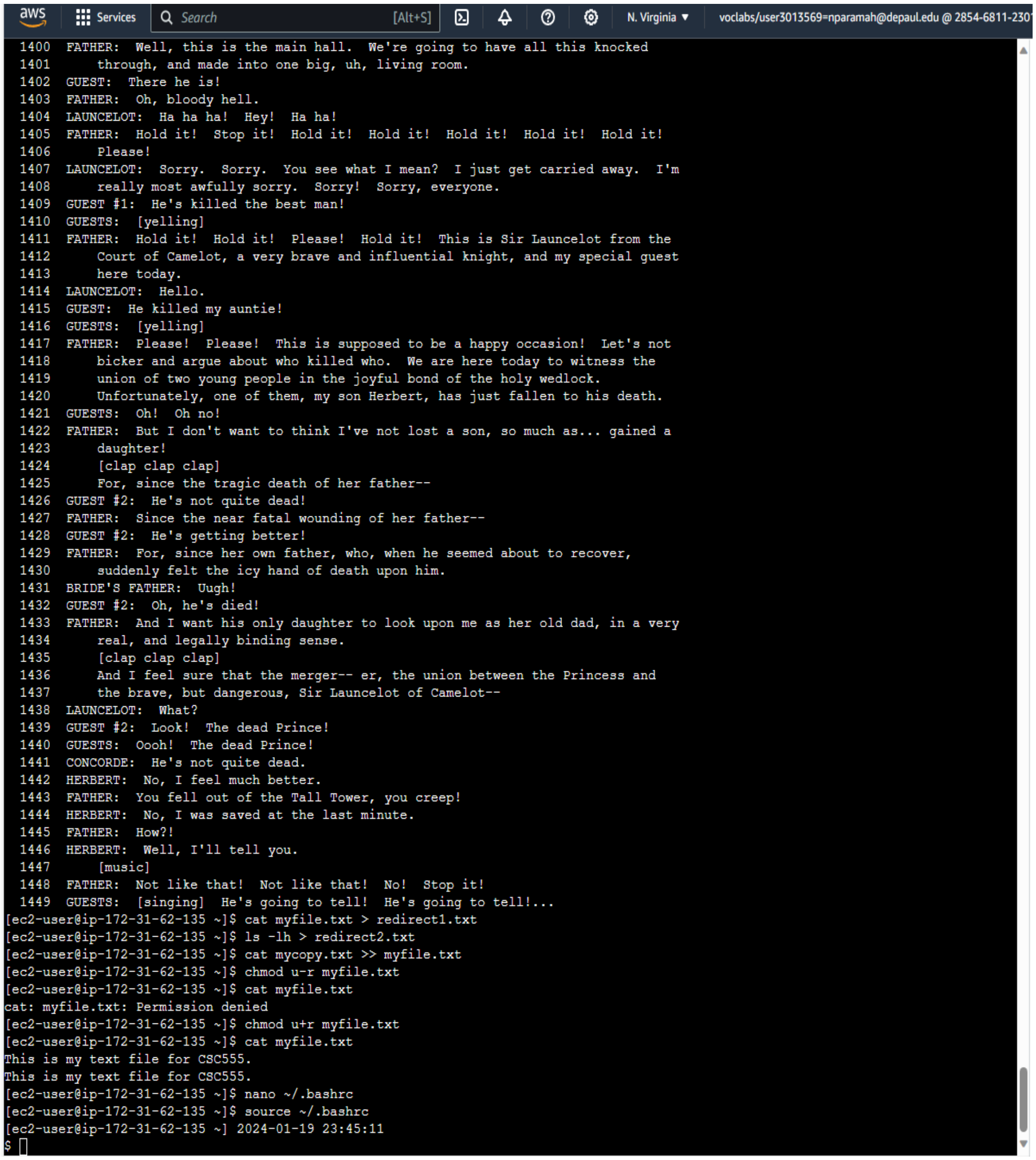
```
[ec2-user@ip-172-31-62-135 ~]$ ls -l
total 88
-rw-r--r--. 1 ec2-user ec2-user 74635 Aug  9  2000 grail
-rw-r--r--. 1 ec2-user ec2-user    33 Jan 19 23:08 mycopy.txt
-rw-r--r--. 1 ec2-user ec2-user    33 Jan 19 23:05 myfile.txt
-rw-r--r--. 1 ec2-user ec2-user   384 Jan 19 23:17 myzipfile.zip
[ec2-user@ip-172-31-62-135 ~]$ ls -lh
total 88K
-rw-r--r--. 1 ec2-user ec2-user 73K Aug  9  2000 grail
-rw-r--r--. 1 ec2-user ec2-user  33 Jan 19 23:08 mycopy.txt
-rw-r--r--. 1 ec2-user ec2-user  33 Jan 19 23:05 myfile.txt
-rw-r--r--. 1 ec2-user ec2-user 384 Jan 19 23:17 myzipfile.zip
[ec2-user@ip-172-31-62-135 ~]$
```

1E



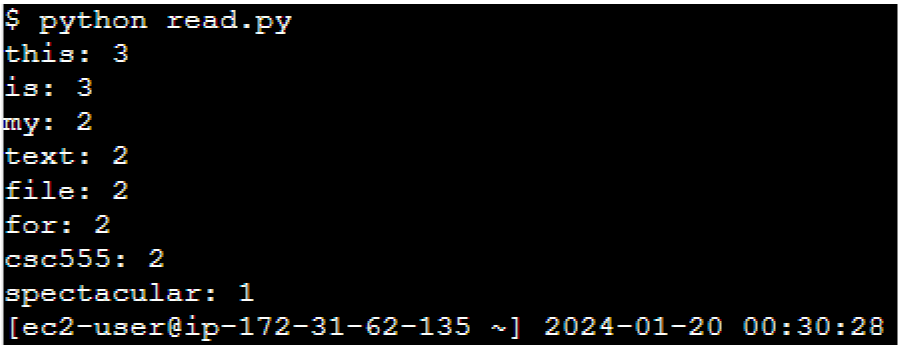
```
[ec2-user@ip-172-31-62-135 ~]$ chmod u-r myfile.txt
[ec2-user@ip-172-31-62-135 ~]$ cat myfile.txt
cat: myfile.txt: Permission denied
```

1F

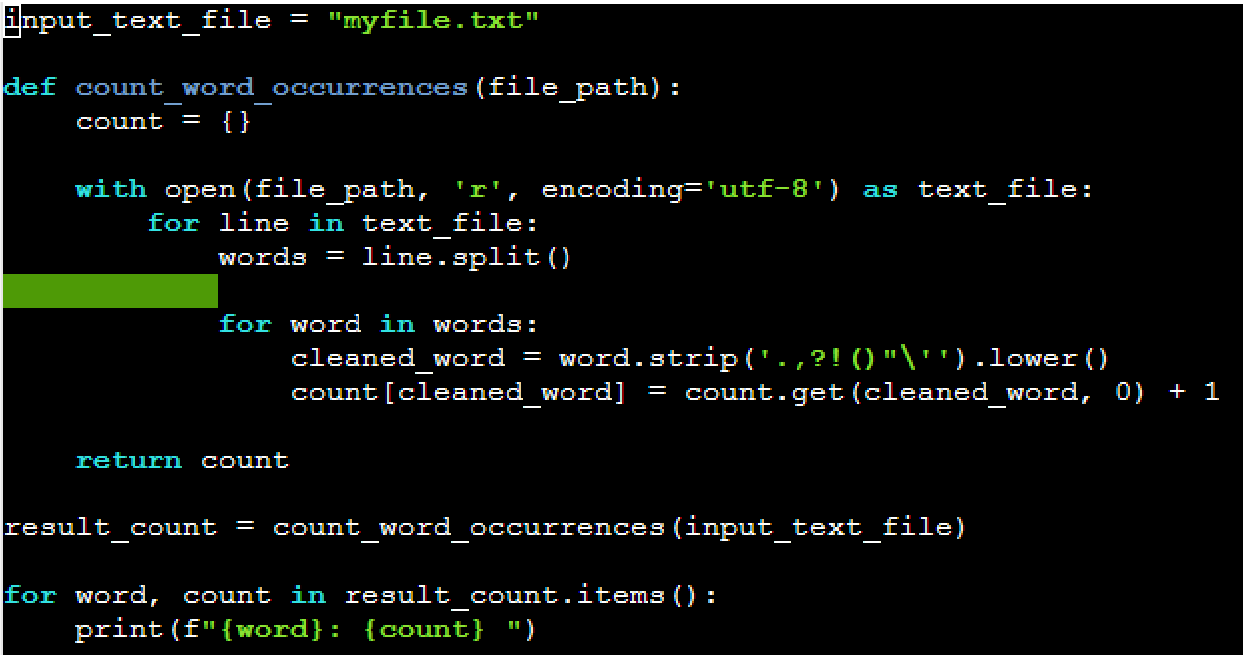


```
 1400  FATHER:  Well, this is the main hall.  We're going to have all this knocked
 1401      through, and made into one big, uh, living room.
 1402  GUEST:  There he is!
 1403  FATHER:  Oh, bloody hell.
 1404  LAUNCELOT:  Ha ha ha!  Hey!  Ha ha!
 1405  FATHER:  Hold it!  Stop it!  Hold it!  Hold it!  Hold it!  Hold it!  Hold it!
 1406      Please!
 1407  LAUNCELOT:  Sorry.  Sorry.  You see what I mean?  I just get carried away.  I'm
 1408      really most awfully sorry.  Sorry!  Sorry, everyone.
 1409  GUEST #1:  He's killed the best man!
 1410  GUESTS:  [yelling]
 1411  FATHER:  Hold it!  Hold it!  Please!  Hold it!  This is Sir Launcelot from the
 1412      Court of Camelot, a very brave and influential knight, and my special guest
 1413      here today.
 1414  LAUNCELOT:  Hello.
 1415  GUEST:  He killed my auntie!
 1416  GUESTS:  [yelling]
 1417  FATHER:  Please!  Please!  This is supposed to be a happy occasion!  Let's not
 1418      bicker and argue about who killed who.  We are here today to witness the
 1419      union of two young people in the joyful bond of the holy wedlock.
 1420      Unfortunately, one of them, my son Herbert, has just fallen to his death.
 1421  GUESTS:  Oh!  Oh no!
 1422  FATHER:  But I don't want to think I've not lost a son, so much as... gained a
 1423      daughter!
 1424      [clap clap clap]
 1425      For, since the tragic death of her father--
 1426  GUEST #2:  He's not quite dead!
 1427  FATHER:  Since the near fatal wounding of her father--
 1428  GUEST #2:  He's getting better!
 1429  FATHER:  For, since her own father, who, when he seemed about to recover,
 1430      suddenly felt the icy hand of death upon him.
 1431  BRIDE'S FATHER:  Uugh!
 1432  GUEST #2:  Oh, he's died!
 1433  FATHER:  And I want his only daughter to look upon me as her old dad, in a very
 1434      real, and legally binding sense.
 1435      [clap clap clap]
 1436      And I feel sure that the merger-- er, the union between the Princess and
 1437      the brave, but dangerous, Sir Launcelot of Camelot--
 1438  LAUNCELOT:  What?
 1439  GUEST #2:  Look!  The dead Prince!
 1440  GUESTS:  Oooh!  The dead Prince!
 1441  CONCORDE:  He's not quite dead.
 1442  HERBERT:  No, I feel much better.
 1443  FATHER:  You fell out of the Tall Tower, you creep!
 1444  HERBERT:  No, I was saved at the last minute.
 1445  FATHER:  How?!
 1446  HERBERT:  Well, I'll tell you.
 1447      [music]
 1448  FATHER:  Not like that!  Not like that!  No!  Stop it!
 1449  GUESTS:  [singing]  He's going to tell!  He's going to tell!...
[ec2-user@ip-172-31-62-135 ~]$ cat myfile.txt > redirect1.txt
[ec2-user@ip-172-31-62-135 ~]$ ls -lh > redirect2.txt
[ec2-user@ip-172-31-62-135 ~]$ cat mycopy.txt >> myfile.txt
[ec2-user@ip-172-31-62-135 ~]$ chmod u-r myfile.txt
[ec2-user@ip-172-31-62-135 ~]$ cat myfile.txt
cat: myfile.txt: Permission denied
[ec2-user@ip-172-31-62-135 ~]$ chmod u+r myfile.txt
[ec2-user@ip-172-31-62-135 ~]$ cat myfile.txt
This is my text file for CSC555.
This is my text file for CSC555.
[ec2-user@ip-172-31-62-135 ~]$ nano ~/.bashrc
[ec2-user@ip-172-31-62-135 ~]$ source ~/.bashrc
[ec2-user@ip-172-31-62-135 ~] 2024-01-19 23:45:11
$ 
```

1G

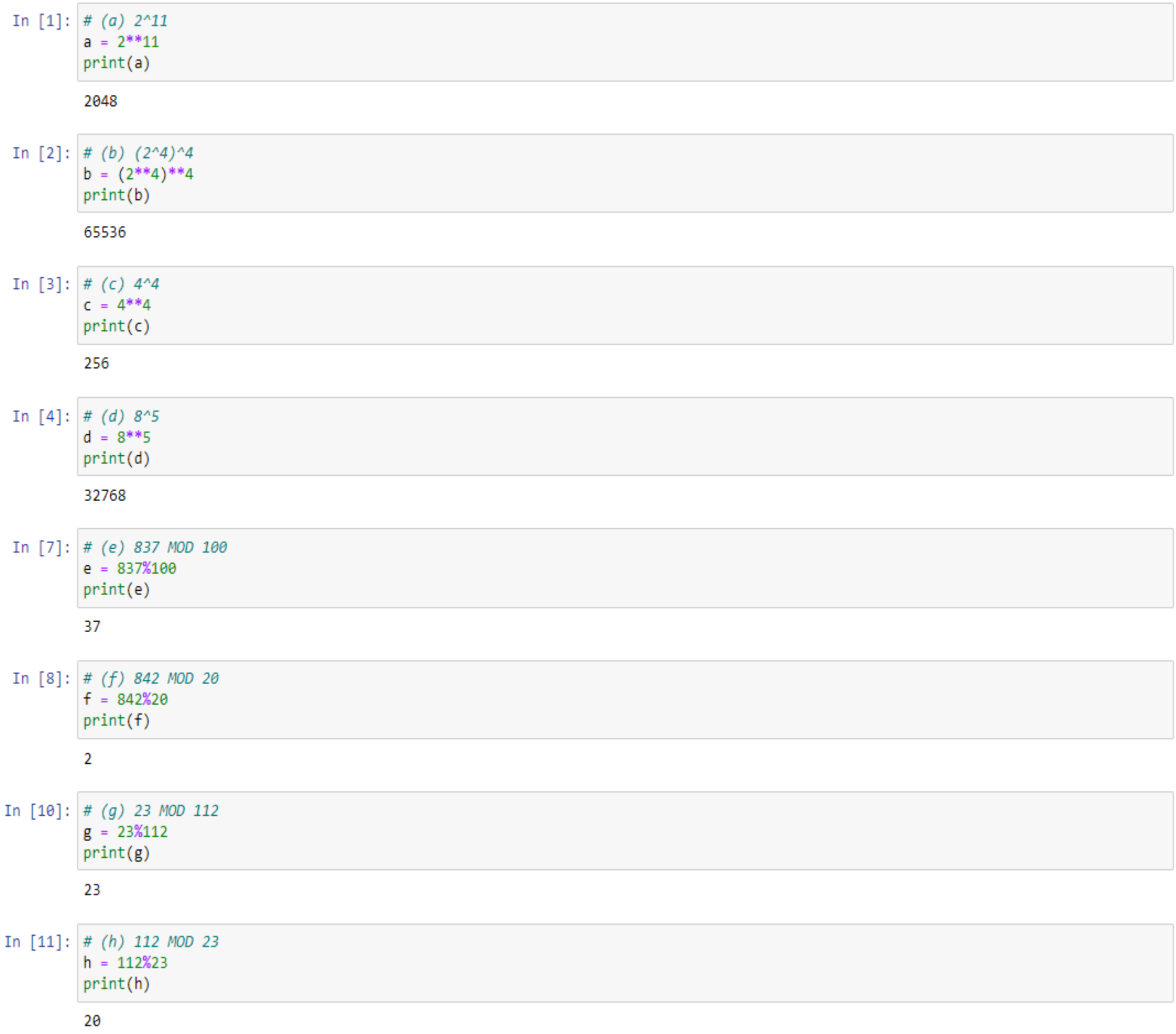


```
$ python read.py
this: 3
is: 3
my: 2
text: 2
file: 2
for: 2
csc555: 2
spectacular: 1
[ec2-user@ip-172-31-62-135 ~] 2024-01-20 00:30:28
```



```
input_text_file = "myfile.txt"

def count_word_occurrences(file_path):
    count = {}

    with open(file_path, 'r', encoding='utf-8') as text_file:
        for line in text_file:
            words = line.split()

            for word in words:
                cleaned_word = word.strip('.,?!()"\'').lower()
                count[cleaned_word] = count.get(cleaned_word, 0) + 1

    return count

result_count = count_word_occurrences(input_text_file)

for word, count in result_count.items():
    print(f"{word}: {count} ")
```

2. **Review: Compute (you can use any tool or software to compute answers in this part – but if you do not know to perform this computation, please talk to me about your course prerequisites):**
   - **a.** 2^{11}
   - **b.** $(2^4)^4$
   - **c.** 4^4
   - **d.** 8^5
   - **e.** 837 MOD 100 (MOD is the modulo operator, a.k.a. the remainder)
   - **f.** 842 MOD 20
   - **g.** 23 MOD 112
   - **h.** 112 MOD 23

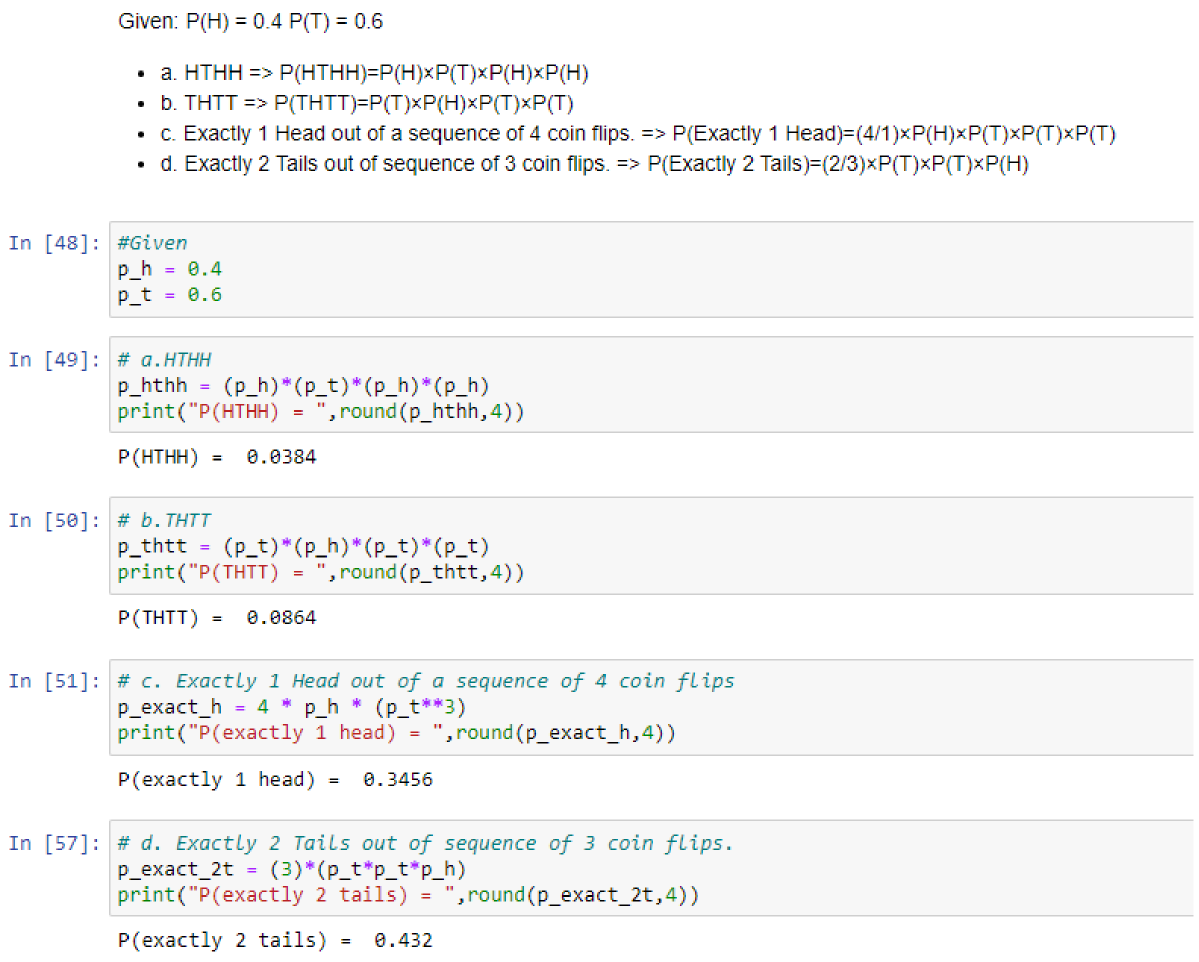


```
In [1]: # (a) 2^11
        a = 2**11
        print(a)

        2048

In [2]: # (b) (2^4)^4
        b = (2**4)**4
        print(b)

        65536

In [3]: # (c) 4^4
        c = 4**4
        print(c)

        256

In [4]: # (d) 8^5
        d = 8**5
        print(d)

        32768

In [7]: # (e) 837 MOD 100
        e = 837%100
        print(e)

        37

In [8]: # (f) 842 MOD 20
        f = 842%20
        print(f)

        2

In [10]: # (g) 23 MOD 112
         g = 23%112
         print(g)

         23

In [11]: # (h) 112 MOD 23
         h = 112%23
         print(h)

         20
```

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

Given:
P(H) = 0.4
P(T) = 0.6

- **a.** HTHH
- **b.** THTT
- **c.** Exactly 1 Head out of a sequence of 4 coin flips.
- **d.** Exactly 2 Tails out of sequence of 3 coin flips.



Given: P(H) = 0.4 P(T) = 0.6

- a. HTHH => P(HTHH)=P(H)×P(T)×P(H)×P(H)
- b. THTT => P(THTT)=P(T)×P(H)×P(T)×P(T)
- c. Exactly 1 Head out of a sequence of 4 coin flips. => P(Exactly 1 Head)=(4/1)×P(H)×P(T)×P(T)×P(T)
- d. Exactly 2 Tails out of sequence of 3 coin flips. => P(Exactly 2 Tails)=(2/3)×P(T)×P(T)×P(H)

In [48]:

```
#Given
p_h = 0.4
p_t = 0.6
```

In [49]:

```
# a.HTHH
p_hthh = (p_h)*(p_t)*(p_h)*(p_h)
print("P(HTHH) = ",round(p_hthh,4))
```

```
P(HTHH) =  0.0384
```

In [50]:

```
# b.THTT
p_thtt = (p_t)*(p_h)*(p_t)*(p_t)
print("P(THTT) = ",round(p_thtt,4))
```

```
P(THTT) =  0.0864
```

In [51]:

```
# c. Exactly 1 Head out of a sequence of 4 coin flips
p_exact_h = 4 * p_h * (p_t**3)
print("P(exactly 1 head) = ",round(p_exact_h,4))
```

```
P(exactly 1 head) =  0.3456
```

In [57]:

```
# d. Exactly 2 Tails out of sequence of 3 coin flips.
p_exact_2t = (3)*(p_t*p_t*p_h)
print("P(exactly 2 tails) = ",round(p_exact_2t,4))
```

```
P(exactly 2 tails) =  0.432
```

4. **Probability Review: Ashley and Allison want to go to the movies to see either Batman or Superman. They are willing to separate and go to different movies. Each of them independently flips a fair, standard coin to decide which movie they will go to. (a) What is the probability that they go to the same movie?**

There are four possible outcomes that Ashley and Allison can have:

1. Batman Batman
2. Batman Superman
3. Superman Batman
4. Superman Superman

The probability of each individual event is P(individual) = ½

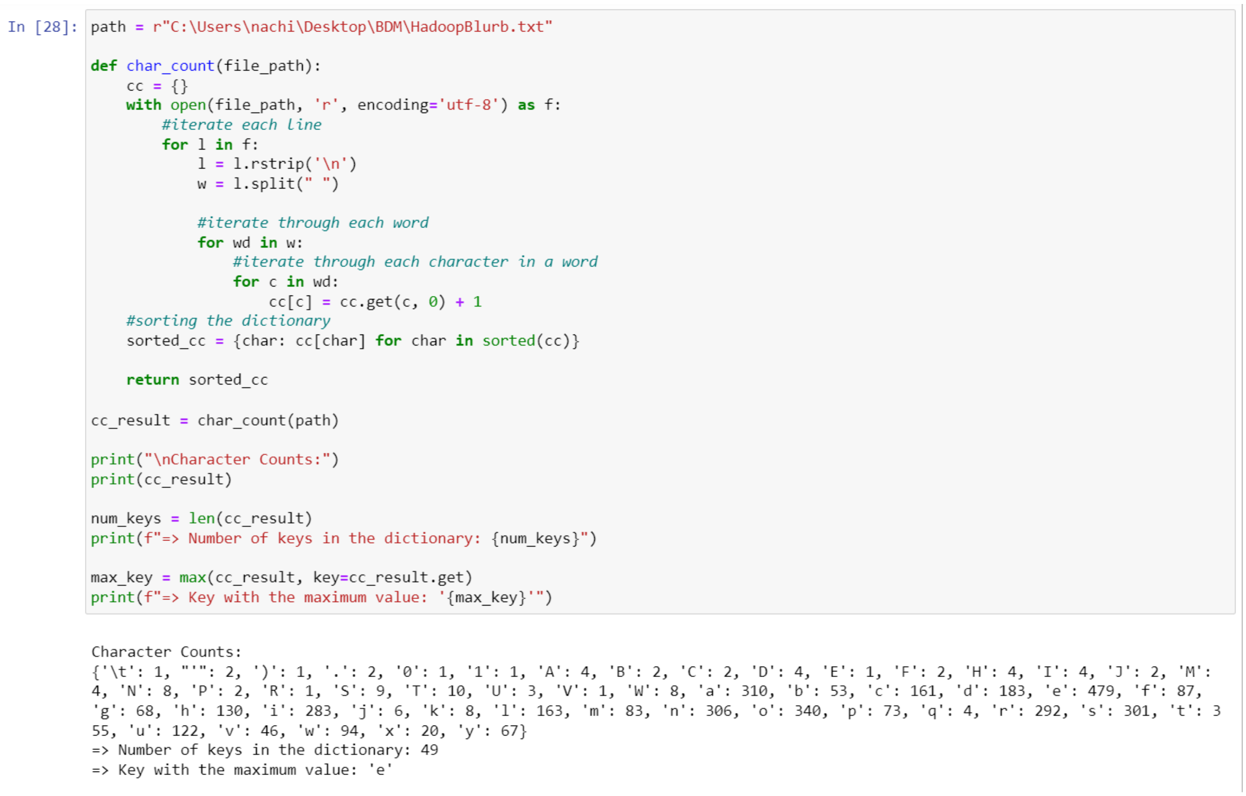
P(Same movie) = P(BB) + P(SS)

P(BB or SS) = P(B) x P(Individual) + P(S) x P(Individual)

P(Same movie) = ½* ½ + ½* ½

P(Same movie) = ½

5. **Python Review This question will review your understanding of Python dictionaries. Write Python code for each part described below and then answer the following questions.**
    a. **Write python code that is going to read a text file (HadoopBlurb.txt on D2L) and compute a total character count using a dictionary (e.g., ‘a’:3, ‘b.’: 2, ‘c’:4 . . . ‘A’:4). For our purposes, a word is anything split by space (.split(‘ ‘)), even if it includes things like punctuation. Characters in a word are contiguous. Note, the above means you are not counting spaces.**

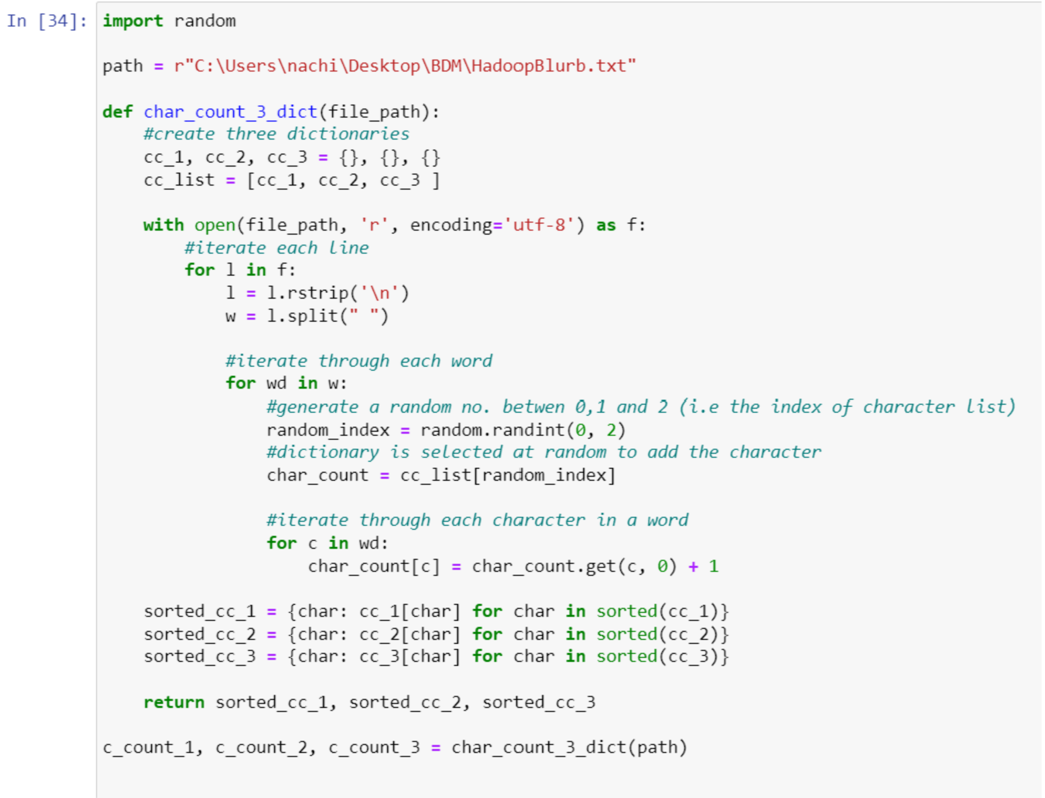


In [28]:

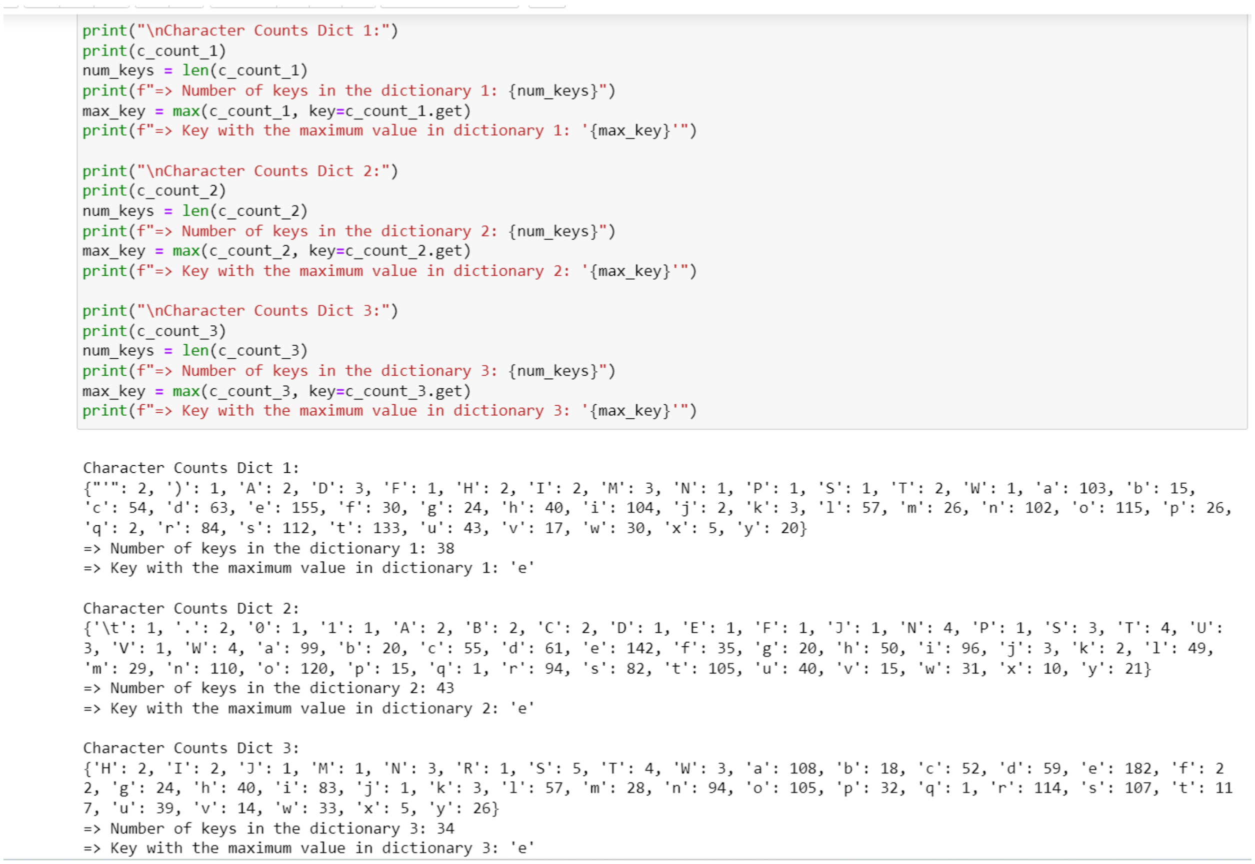
```python
path = r"C:\Users\nachi\Desktop\BDM\HadoopBlurb.txt"

def char_count(file_path):
    cc = {}
    with open(file_path, 'r', encoding='utf-8') as f:
        #iterate each line
        for l in f:
            l = l.rstrip('\n')
            w = l.split(" ")

            #iterate through each word
            for wd in w:
                #iterate through each character in a word
                for c in wd:
                    cc[c] = cc.get(c, 0) + 1
    #sorting the dictionary
    sorted_cc = {char: cc[char] for char in sorted(cc)}

    return sorted_cc

cc_result = char_count(path)

print("\nCharacter Counts:")
print(cc_result)

num_keys = len(cc_result)
print(f"=> Number of keys in the dictionary: {num_keys}")

max_key = max(cc_result, key=cc_result.get)
print(f"=> Key with the maximum value: '{max_key}'")
```

```
Character Counts:
{'\t': 1, "’": 2, ')': 1, '.': 2, '0': 1, '1': 1, 'A': 4, 'B': 2, 'C': 2, 'D': 4, 'E': 1, 'F': 2, 'H': 4, 'I': 4, 'J': 2, 'M':
4, 'N': 8, 'P': 2, 'R': 1, 'S': 9, 'T': 10, 'U': 3, 'V': 1, 'W': 8, 'a': 310, 'b': 53, 'c': 161, 'd': 183, 'e': 479, 'f': 87,
'g': 68, 'h': 130, 'i': 283, 'j': 6, 'k': 8, 'l': 163, 'm': 83, 'n': 306, 'o': 340, 'p': 73, 'q': 4, 'r': 292, 's': 301, 't': 3
55, 'u': 122, 'v': 46, 'w': 94, 'x': 20, 'y': 67}
=> Number of keys in the dictionary: 49
=> Key with the maximum value: 'e'
```

    b. **Write python code that is going to create three different character count dictionaries instead, assigning the characters at random. Each time you process a word, choose at random which count dictionary to add it to and then add all characters of that word to that dictionary (that means characters of some words will appear in all three dictionaries simultaneously).**



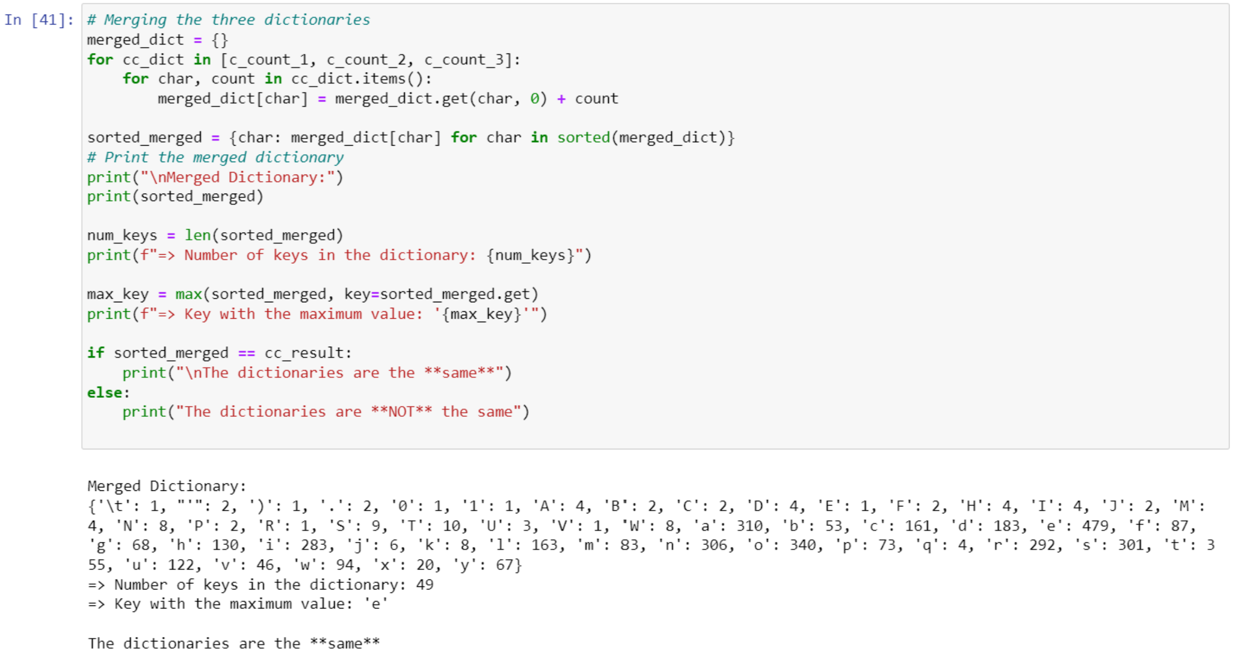
```
In [34]: import random

         path = r"C:\Users\nachi\Desktop\BDM\HadoopBlurb.txt"

         def char_count_3_dict(file_path):
             #create three dictionaries
             cc_1, cc_2, cc_3 = {}, {}, {}
             cc_list = [cc_1, cc_2, cc_3 ]

             with open(file_path, 'r', encoding='utf-8') as f:
                 #iterate each line
                 for l in f:
                     l = l.rstrip('\n')
                     w = l.split(" ")

                     #iterate through each word
                     for wd in w:
                         #generate a random no. betwen 0,1 and 2 (i.e the index of character list)
                         random_index = random.randint(0, 2)
                         #dictionary is selected at random to add the character
                         char_count = cc_list[random_index]

                         #iterate through each character in a word
                         for c in wd:
                             char_count[c] = char_count.get(c, 0) + 1

             sorted_cc_1 = {char: cc_1[char] for char in sorted(cc_1)}
             sorted_cc_2 = {char: cc_2[char] for char in sorted(cc_2)}
             sorted_cc_3 = {char: cc_3[char] for char in sorted(cc_3)}

             return sorted_cc_1, sorted_cc_2, sorted_cc_3

         c_count_1, c_count_2, c_count_3 = char_count_3_dict(path)
```



```
print("\nCharacter Counts Dict 1:")
print(c_count_1)
num_keys = len(c_count_1)
print(f"=> Number of keys in the dictionary 1: {num_keys}")
max_key = max(c_count_1, key=c_count_1.get)
print(f"=> Key with the maximum value in dictionary 1: '{max_key}'")

print("\nCharacter Counts Dict 2:")
print(c_count_2)
num_keys = len(c_count_2)
print(f"=> Number of keys in the dictionary 2: {num_keys}")
max_key = max(c_count_2, key=c_count_2.get)
print(f"=> Key with the maximum value in dictionary 2: '{max_key}'")

print("\nCharacter Counts Dict 3:")
print(c_count_3)
num_keys = len(c_count_3)
print(f"=> Number of keys in the dictionary 3: {num_keys}")
max_key = max(c_count_3, key=c_count_3.get)
print(f"=> Key with the maximum value in dictionary 3: '{max_key}'")
```

```
Character Counts Dict 1:
{"'": 2, ')': 1, 'A': 2, 'D': 3, 'F': 1, 'H': 2, 'I': 2, 'M': 3, 'N': 1, 'P': 1, 'S': 1, 'T': 2, 'W': 1, 'a': 103, 'b': 15,
'c': 54, 'd': 63, 'e': 155, 'f': 30, 'g': 24, 'h': 40, 'i': 104, 'j': 2, 'k': 3, 'l': 57, 'm': 26, 'n': 102, 'o': 115, 'p': 26,
'q': 2, 'r': 84, 's': 112, 't': 133, 'u': 43, 'v': 17, 'w': 30, 'x': 5, 'y': 20}
=> Number of keys in the dictionary 1: 38
=> Key with the maximum value in dictionary 1: 'e'

Character Counts Dict 2:
{'\t': 1, '.': 2, '0': 1, '1': 1, 'A': 2, 'B': 2, 'C': 2, 'D': 1, 'E': 1, 'F': 1, 'J': 1, 'N': 4, 'P': 1, 'S': 3, 'T': 4, 'U':
3, 'V': 1, 'W': 4, 'a': 99, 'b': 20, 'c': 55, 'd': 61, 'e': 142, 'f': 35, 'g': 20, 'h': 50, 'i': 96, 'j': 3, 'k': 2, 'l': 49,
'm': 29, 'n': 110, 'o': 120, 'p': 15, 'q': 1, 'r': 94, 's': 82, 't': 105, 'u': 40, 'v': 15, 'w': 31, 'x': 10, 'y': 21}
=> Number of keys in the dictionary 2: 43
=> Key with the maximum value in dictionary 2: 'e'

Character Counts Dict 3:
{'H': 2, 'I': 2, 'J': 1, 'M': 1, 'N': 3, 'R': 1, 'S': 5, 'T': 4, 'W': 3, 'a': 108, 'b': 18, 'c': 52, 'd': 59, 'e': 182, 'f': 2
2, 'g': 24, 'h': 40, 'i': 83, 'j': 1, 'k': 3, 'l': 57, 'm': 28, 'n': 94, 'o': 105, 'p': 32, 'q': 1, 'r': 114, 's': 107, 't': 11
7, 'u': 39, 'v': 14, 'w': 33, 'x': 5, 'y': 26}
=> Number of keys in the dictionary 3: 34
=> Key with the maximum value in dictionary 3: 'e'
```

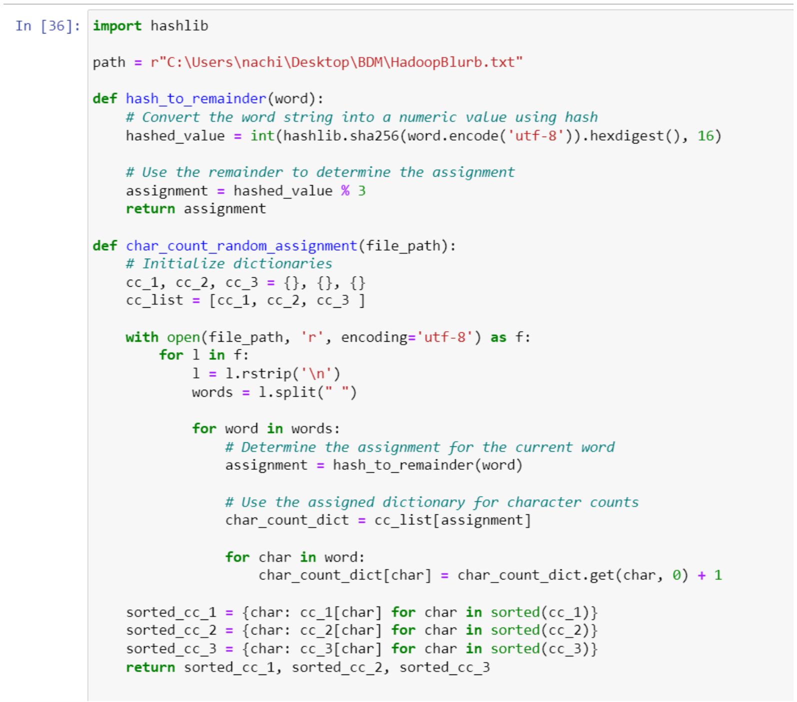
c. **Write python code to merge the three dictionaries into one (adding the counts) and verify that it matches the dictionary from Part 5-a.**



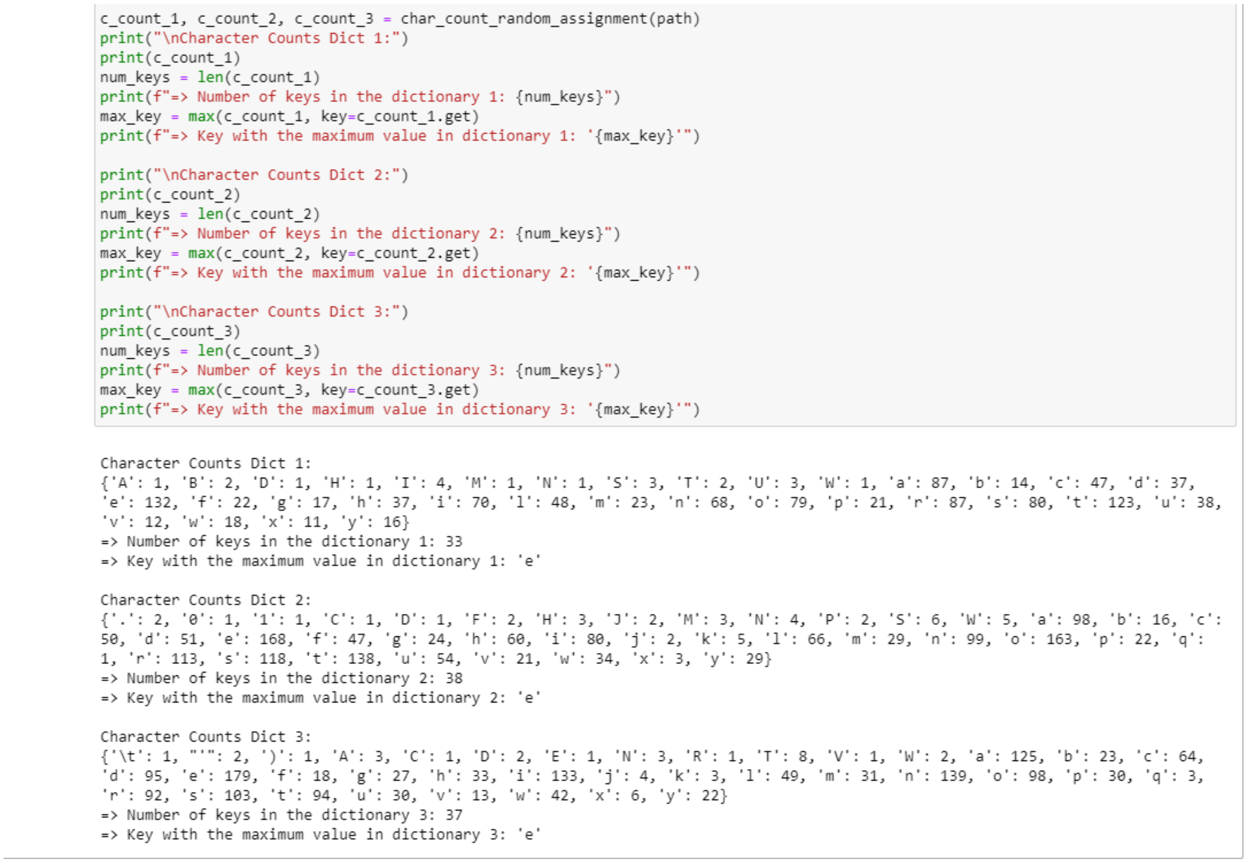
```
In [41]: # Merging the three dictionaries
         merged_dict = {}
         for cc_dict in [c_count_1, c_count_2, c_count_3]:
             for char, count in cc_dict.items():
                 merged_dict[char] = merged_dict.get(char, 0) + count

         sorted_merged = {char: merged_dict[char] for char in sorted(merged_dict)}
         # Print the merged dictionary
         print("\nMerged Dictionary:")
         print(sorted_merged)

         num_keys = len(sorted_merged)
         print(f"=> Number of keys in the dictionary: {num_keys}")

         max_key = max(sorted_merged, key=sorted_merged.get)
         print(f"=> Key with the maximum value: '{max_key}'")

         if sorted_merged == cc_result:
             print("\nThe dictionaries are the **same**")
         else:
             print("The dictionaries are **NOT** the same")
```

```
Merged Dictionary:
{'\t': 1, "'": 2, ')': 1, '.': 2, '0': 1, '1': 1, 'A': 4, 'B': 2, 'C': 2, 'D': 4, 'E': 1, 'F': 2, 'H': 4, 'I': 4, 'J': 2, 'M':
4, 'N': 8, 'P': 2, 'R': 1, 'S': 9, 'T': 10, 'U': 3, 'V': 1, 'W': 8, 'a': 310, 'b': 53, 'c': 161, 'd': 183, 'e': 479, 'f': 87,
'g': 68, 'h': 130, 'i': 283, 'j': 6, 'k': 8, 'l': 163, 'm': 83, 'n': 306, 'o': 340, 'p': 73, 'q': 4, 'r': 292, 's': 301, 't': 3
55, 'u': 122, 'v': 46, 'w': 94, 'x': 20, 'y': 67}
=> Number of keys in the dictionary: 49
=> Key with the maximum value: 'e'

The dictionaries are the **same**
```

d. **Write python code that is going to randomly but deterministically assign each word to one of the three dictionaries instead. For example, you can make that assignment using the remainder (YourNumber % 3 will always return 0 or 1 or 2 depending on the number). You can convert a word string into a numeric value using hash (e.g., hash('Hadoop.')).**



```
In [36]: import hashlib

         path = r"C:\Users\nachi\Desktop\BDM\HadoopBlurb.txt"

         def hash_to_remainder(word):
             # Convert the word string into a numeric value using hash
             hashed_value = int(hashlib.sha256(word.encode('utf-8')).hexdigest(), 16)

             # Use the remainder to determine the assignment
             assignment = hashed_value % 3
             return assignment

         def char_count_random_assignment(file_path):
             # Initialize dictionaries
             cc_1, cc_2, cc_3 = {}, {}, {}
             cc_list = [cc_1, cc_2, cc_3 ]

             with open(file_path, 'r', encoding='utf-8') as f:
                 for l in f:
                     l = l.rstrip('\n')
                     words = l.split(" ")

                     for word in words:
                         # Determine the assignment for the current word
                         assignment = hash_to_remainder(word)

                         # Use the assigned dictionary for character counts
                         char_count_dict = cc_list[assignment]

                         for char in word:
                             char_count_dict[char] = char_count_dict.get(char, 0) + 1

             sorted_cc_1 = {char: cc_1[char] for char in sorted(cc_1)}
             sorted_cc_2 = {char: cc_2[char] for char in sorted(cc_2)}
             sorted_cc_3 = {char: cc_3[char] for char in sorted(cc_3)}
             return sorted_cc_1, sorted_cc_2, sorted_cc_3
```



```
c_count_1, c_count_2, c_count_3 = char_count_random_assignment(path)
print("\nCharacter Counts Dict 1:")
print(c_count_1)
num_keys = len(c_count_1)
print(f"=> Number of keys in the dictionary 1: {num_keys}")
max_key = max(c_count_1, key=c_count_1.get)
print(f"=> Key with the maximum value in dictionary 1: '{max_key}'")

print("\nCharacter Counts Dict 2:")
print(c_count_2)
num_keys = len(c_count_2)
print(f"=> Number of keys in the dictionary 2: {num_keys}")
max_key = max(c_count_2, key=c_count_2.get)
print(f"=> Key with the maximum value in dictionary 2: '{max_key}'")

print("\nCharacter Counts Dict 3:")
print(c_count_3)
num_keys = len(c_count_3)
print(f"=> Number of keys in the dictionary 3: {num_keys}")
max_key = max(c_count_3, key=c_count_3.get)
print(f"=> Key with the maximum value in dictionary 3: '{max_key}'")
```

```
Character Counts Dict 1:
{'A': 1, 'B': 2, 'D': 1, 'H': 1, 'I': 4, 'M': 1, 'N': 1, 'S': 3, 'T': 2, 'U': 3, 'W': 1, 'a': 87, 'b': 14, 'c': 47, 'd': 37,
'e': 132, 'f': 22, 'g': 17, 'h': 37, 'i': 70, 'l': 48, 'm': 23, 'n': 68, 'o': 79, 'p': 21, 'r': 87, 's': 80, 't': 123, 'u': 38,
'v': 12, 'w': 18, 'x': 11, 'y': 16}
=> Number of keys in the dictionary 1: 33
=> Key with the maximum value in dictionary 1: 'e'

Character Counts Dict 2:
{'.': 2, '0': 1, '1': 1, 'C': 1, 'D': 1, 'F': 2, 'H': 3, 'J': 2, 'M': 3, 'N': 4, 'P': 2, 'S': 6, 'W': 5, 'a': 98, 'b': 16, 'c':
50, 'd': 51, 'e': 168, 'f': 47, 'g': 24, 'h': 60, 'i': 80, 'j': 2, 'k': 5, 'l': 66, 'm': 29, 'n': 99, 'o': 163, 'p': 22, 'q':
1, 'r': 113, 's': 118, 't': 138, 'u': 54, 'v': 21, 'w': 34, 'x': 3, 'y': 29}
=> Number of keys in the dictionary 2: 38
=> Key with the maximum value in dictionary 2: 'e'

Character Counts Dict 3:
{'\t': 1, "'": 2, ')': 1, 'A': 3, 'C': 1, 'D': 2, 'E': 1, 'N': 3, 'R': 1, 'T': 8, 'V': 1, 'W': 2, 'a': 125, 'b': 23, 'c': 64,
'd': 95, 'e': 179, 'f': 18, 'g': 27, 'h': 33, 'i': 133, 'j': 4, 'k': 3, 'l': 49, 'm': 31, 'n': 139, 'o': 98, 'p': 30, 'q': 3,
'r': 92, 's': 103, 't': 94, 'u': 30, 'v': 13, 'w': 42, 'x': 6, 'y': 22}
=> Number of keys in the dictionary 3: 37
=> Key with the maximum value in dictionary 3: 'e'
```

6. **HDFS. Compute or explain the following (in a few sentences):**
   a. **What are the guarantees offered by a replication factor of 3 (3 copies of each block)?**

HDFS replication factor of three means that every block is stored three times on distinct nodes in the Hadoop cluster. This procedure offers tolerance to faults and resilience against node failures. The ability of the system to preserve data availability and dependability even in the case of a node failure is demonstrated by the fact that the data can still be retrieved and accessed from the surviving copies.

   b. **What action does NameNode have to take when a machine in the Hadoop cluster fails/crashes?**

The NameNode is in charge of identifying failures or crashes on Hadoop cluster machines. The NameNode starts the replication process to make more copies of the lost data blocks on other functioning nodes in the cluster as soon as it detects a problem.

   c. **Give an HDFS block size of 256MB, how many blocks will be allocated for a file size of 5GB.**

No. of blocks = File Size/Block size = ((5*1024)/256) = **20 Blocks**

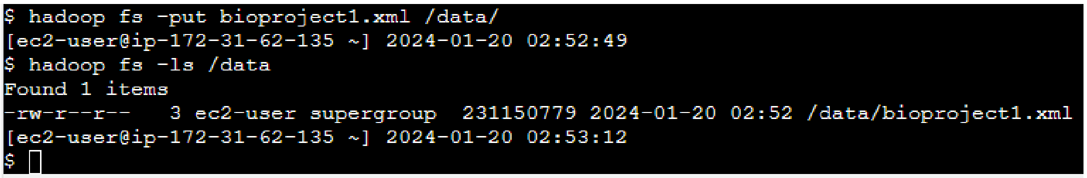
   d. **What is the overall storage cost for a file of size 950 MBs, when the HDFS replication factor is set to 4?**

For replication factor of 4 each block is replicated four times.

Cost = 4*950 = 3800mb

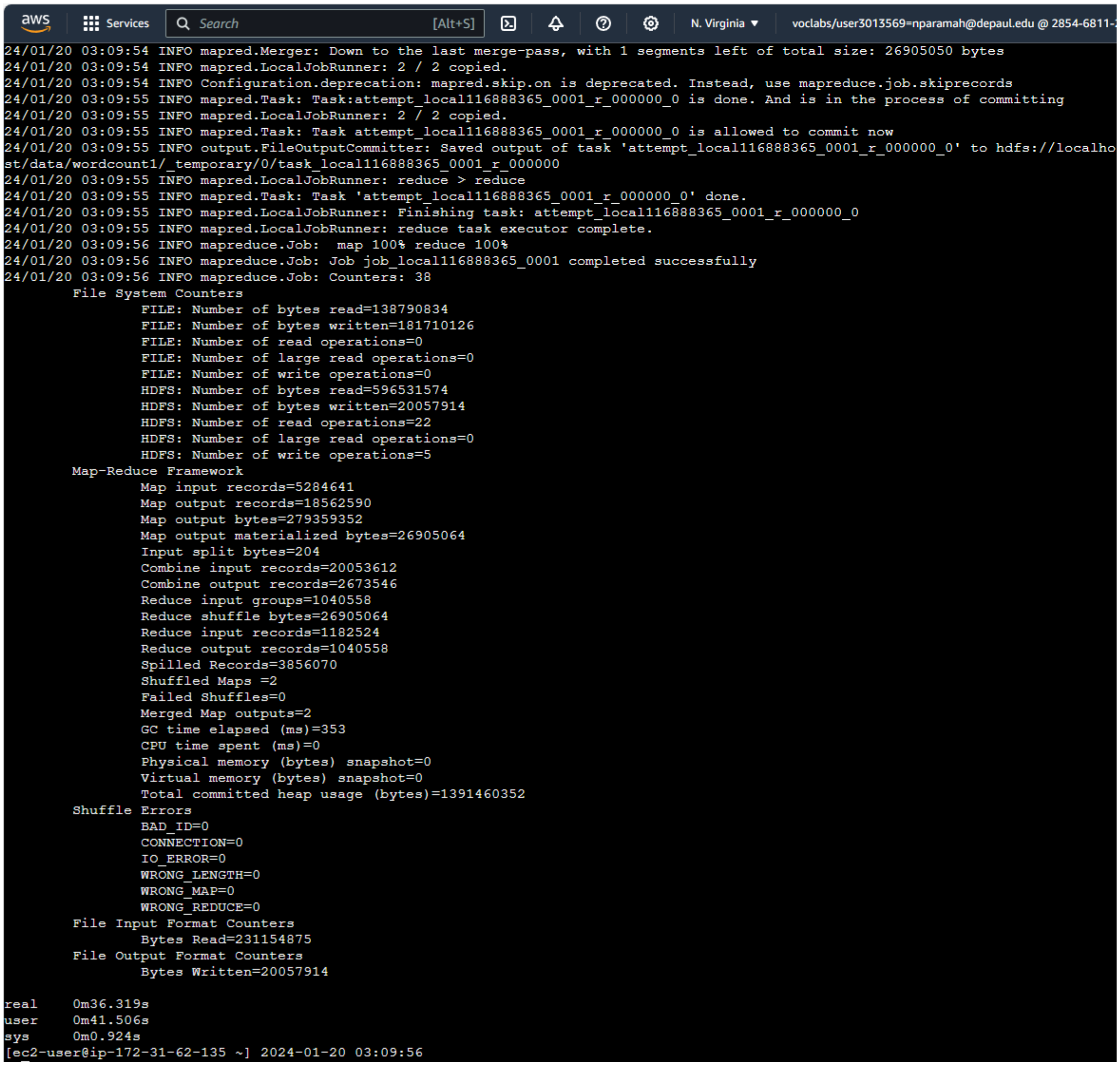
7. **Hadoop Practice For this part of the assignment, you will run wordcount on a single-node Hadoop instance. The instructions provided below are following Hadoop: The Definitive Guide instructions presented in Appendix A: Installing Apache Hadoop.**

7A



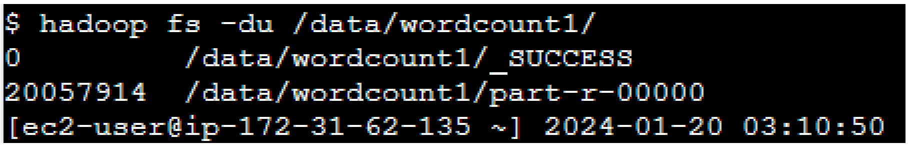
```
$ hadoop fs -put bioproject1.xml /data/
[ec2-user@ip-172-31-62-135 ~] 2024-01-20 02:52:49
$ hadoop fs -ls /data
Found 1 items
-rw-r--r--   3 ec2-user supergroup  231150779 2024-01-20 02:52 /data/bioproject1.xml
[ec2-user@ip-172-31-62-135 ~] 2024-01-20 02:53:12
$ 
```

7B



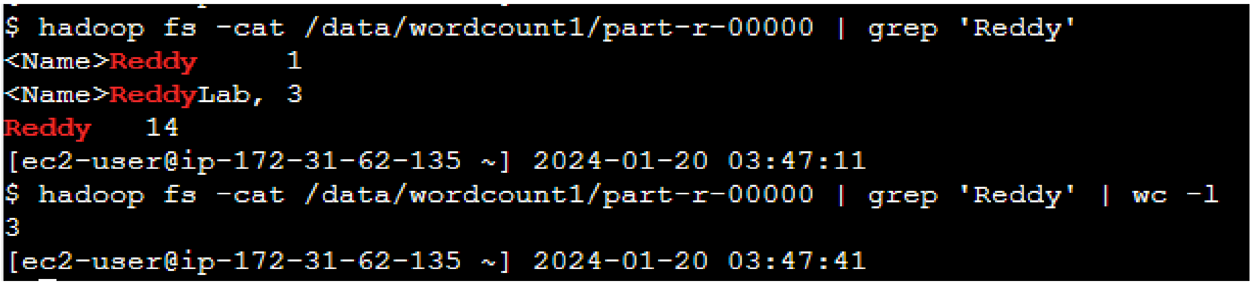
```
24/01/20 03:09:54 INFO mapred.Merger: Down to the last merge-pass, with 1 segments left of total size: 26905050 bytes
24/01/20 03:09:54 INFO mapred.LocalJobRunner: 2 / 2 copied.
24/01/20 03:09:54 INFO Configuration.deprecation: mapred.skip.on is deprecated. Instead, use mapreduce.job.skiprecords
24/01/20 03:09:55 INFO mapred.Task: Task:attempt_local116888365_0001_r_000000_0 is done. And is in the process of committing
24/01/20 03:09:55 INFO mapred.LocalJobRunner: 2 / 2 copied.
24/01/20 03:09:55 INFO mapred.Task: Task attempt_local116888365_0001_r_000000_0 is allowed to commit now
24/01/20 03:09:55 INFO output.FileOutputCommitter: Saved output of task 'attempt_local116888365_0001_r_000000_0' to hdfs://localho
st/data/wordcount1/_temporary/0/task_local116888365_0001_r_000000
24/01/20 03:09:55 INFO mapred.LocalJobRunner: reduce > reduce
24/01/20 03:09:55 INFO mapred.Task: Task 'attempt_local116888365_0001_r_000000_0' done.
24/01/20 03:09:55 INFO mapred.LocalJobRunner: Finishing task: attempt_local116888365_0001_r_000000_0
24/01/20 03:09:55 INFO mapred.LocalJobRunner: reduce task executor complete.
24/01/20 03:09:56 INFO mapreduce.Job:  map 100% reduce 100%
24/01/20 03:09:56 INFO mapreduce.Job: Job job_local116888365_0001 completed successfully
24/01/20 03:09:56 INFO mapreduce.Job: Counters: 38
        File System Counters
                FILE: Number of bytes read=138790834
                FILE: Number of bytes written=181710126
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=596531574
                HDFS: Number of bytes written=20057914
                HDFS: Number of read operations=22
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=5
        Map-Reduce Framework
                Map input records=5284641
                Map output records=18562590
                Map output bytes=279359352
                Map output materialized bytes=26905064
                Input split bytes=204
                Combine input records=20053612
                Combine output records=2673546
                Reduce input groups=1040558
                Reduce shuffle bytes=26905064
                Reduce input records=1182524
                Reduce output records=1040558
                Spilled Records=3856070
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=353
                CPU time spent (ms)=0
                Physical memory (bytes) snapshot=0
                Virtual memory (bytes) snapshot=0
                Total committed heap usage (bytes)=1391460352
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=231154875
        File Output Format Counters
                Bytes Written=20057914

real    0m36.319s
user    0m41.506s
sys     0m0.924s
[ec2-user@ip-172-31-62-135 ~] 2024-01-20 03:09:56
```

7C



```
$ hadoop fs -du /data/wordcount1/
0         /data/wordcount1/_SUCCESS
20057914  /data/wordcount1/part-r-00000
[ec2-user@ip-172-31-62-135 ~] 2024-01-20 03:10:50
```

7D



```
$ hadoop fs -cat /data/wordcount1/part-r-00000 | grep 'Reddy'
<Name>Reddy     1
<Name>ReddyLab, 3
Reddy   14
[ec2-user@ip-172-31-62-135 ~] 2024-01-20 03:47:11
$ hadoop fs -cat /data/wordcount1/part-r-00000 | grep 'Reddy' | wc -l
3
[ec2-user@ip-172-31-62-135 ~] 2024-01-20 03:47:41
```