CSc 3320: Systems Programming

Spring 2021

Homework

2: Total points 100

Submission instructions:

1. Create a Google doc for each homework assignment submission.
2. Start your responses from page 2 of the document and copy these instructions on page 1.
3. Fill in your name, campus ID and panther # in the fields provided. If this information is missing in your document TWO POINTS WILL BE DEDUCTED per submission.
4. Keep this page 1 intact on all your submissions. If this *submissions instructions* page is missing in your submission TWO POINTS WILL BE DEDUCTED per submission.
5. Each homework will typically have 2-3 PARTS, where each PART focuses on specific topic(s).
6. Start your responses to each PART on a new page.
7. If you are being asked to write code copy the code into a separate txt file and submit that as well.
8. If you are being asked to test code or run specific commands or scripts, provide the evidence of your outputs through a screenshot and copy the same into the document.
9. Upon completion, download a .PDF version of the document and submit the same.

Full Name: **Jamie Lopez**

Campus ID: **jrogers75**

Panther #: **001464896**

PART 1 (2.5 points each): 10pts

1. What are the differences among ***grep***, ***egrep*** *and* ***fgrep***? Describe using an example.

   $cat poem.txt (to display sample file)

   Command grep searches for a pattern in a list of files. All matched lines are displayed as standard output. Regular expressions can be part of the search pattern:

   $grep ".immed" poem.txt

```
[jrogers75@gsuad.gsu.edu@snowball Homeworks]$ grep ".immed" poem.txt
And often is his gold complexion dimmed;
By chance, or nature's changing course, untrimmed;
[jrogers75@gsuad.gsu.edu@snowball Homeworks]$
```

   Command egrep supports matching patterns using extended regular expressions:

   $egrep "death | eyes" poem.txt

```
[jrogers75@gsuad.gsu.edu@snowball Homeworks]$ egrep "death | eyes" poem.txt
Nor shall death brag thou wand'rest in his shade,
So long as men can breathe, or eyes can see,
```

   Command fgrep is a version of grep that only searches for a fixed string:

   $fgrep "day?" poem.txt

```
[jrogers75@gsuad.gsu.edu@snowball Homeworks]$ cat poem.txt
Shall I compare thee to a summer's day?
Thou art more lovely and more temperate.
Rough winds do shake the darling buds of May,
And summer's lease hath all too short a date.
Sometime too hot the eye of heaven shines.
And often is his gold complexion dimmed;
And every fair from fair sometime declines,
By chance, or nature's changing course, untrimmed;
But thy eternal summer shall not fade,
Nor lose possession of that fair thou ow'st,
Nor shall death brag thou wand'rest in his shade,
When in eternal lines to Time thou grow'st.
So long as men can breathe, or eyes can see,
So long lives this, and this gives life to thee.

[jrogers75@gsuad.gsu.edu@snowball Homeworks]$ grep ".immed" poem.txt
And often is his gold complexion dimmed;
By chance, or nature's changing course, untrimmed;
[jrogers75@gsuad.gsu.edu@snowball Homeworks]$ egrep "death|eyes" poem.txt
Nor shall death brag thou wand'rest in his shade,
So long as men can breathe, or eyes can see,
[jrogers75@gsuad.gsu.edu@snowball Homeworks]$ fgrep "day?" poem.txt
Shall I compare thee to a summer's day?
```

2. Which utility can be used to compress and decompress files? And how to compress multiple files into a single file? Please provide one example for it.

   The utilities **compress**, **uncompress**, **gzip**, and **gunzip** can be used to compress and uncompress files to help save disk space. Multiple files can be compressed into a single file by using the command **tar** for creating an archive. This command allows you to save directory structures into one single backup volume. Commands **cpio** and **tar** are useful for backing up small subdirectories. Whereas, **dump** is useful for creating large backup files. See mocked up data below for example.

   Create a tar file:

   $tar -cvf poetry.tar poems

   poems/
   poems/favorite.csh
   poems/funny/
   poems/author/shakespeare.csh
   poems/sad.csh
   poems/love.csh

   Show contents in tar file:

   tar -tvf poetry.tar

   drwxr-xr-x jll/jll 0 2021-01-15 11:12 poems/
   -rwxr-xr-x jll/jll 403 2021-02-08 01:23 poems/favorite.csh
   drwxr-xr-x jll/jll 0 2021-23-01 11:26 poems/funny/
   -rwxr-xr-x jll/jll 1475 2021-01-13 04:25 poems/author/shakespeare.csh
   -rwxr-xr-x jll/jll 1475 2021-01-30 03:17 poems/sad.csh
   -rwxr-xr-x jll/jll 1475 2021-01-12 01:45 poems/love.csh

3. Which utility (or utilities) can break a line into multiple fields by defining a separator? What is the default separator? How to define a separator manually in the command line? Please provide one example for defining the separator for each utility.

   The command **sort** can define a separator by using **-t** and **awk** can define a separator by using **-F**. The default separator is a blank space - tabs and/or spaces.

   Examples: sort - used -r to reverse sort -t: for separator by colon so second column has been sorted in reverse order

```
[jrogers75@gsuad.gsu.edu@snowball Homeworks]$ cat samplecolon.txt
Ocean's: 11: First: Movie
Ocean's: 12: Second: Movie
Ocean's: 13: Third: Movie

[jrogers75@gsuad.gsu.edu@snowball Homeworks]$ sort -r -t: +1 -2 samplecolon.txt
Ocean's: 13: Third: Movie
Ocean's: 12: Second: Movie
Ocean's: 11: First: Movie
```

   awk: used awk with -F: to display second column which is numbers

```
[jrogers75@gsuad.gsu.edu@snowball Homeworks]$ cat samplecolon.txt
Ocean's: 11: First: Movie
Ocean's: 12: Second: Movie
Ocean's: 13: Third: Movie

[jrogers75@gsuad.gsu.edu@snowball Homeworks]$ awk -F: '{print $2}' samplecolon.
txt
 11
 12
 13
```

4. What does the ***sort*** command do? What are the different possible fields? Explain using an example.

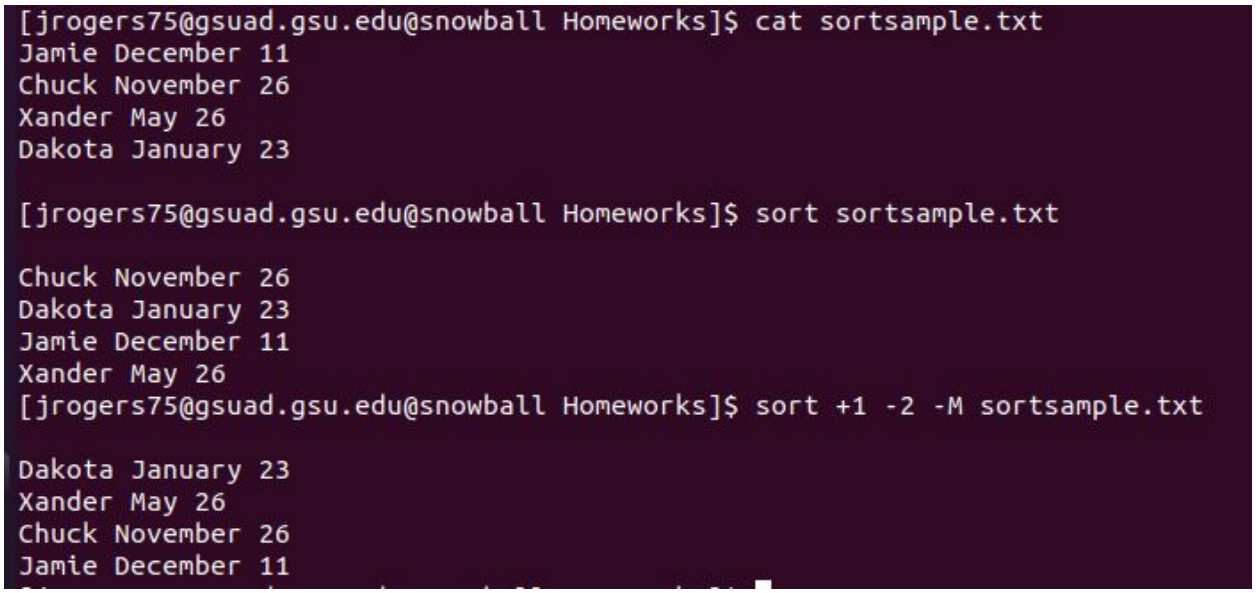
   The sort command sorts files in ascending or descending order based on one or more sort fields. The default sort is ascending. Individual fields are ordered lexicographically

(corresponds to the ASCII character value). All fields of a line are considered for sorting unless one or more sort fields are specified. Specific fields can be sorted such as the first field, the third field, etc. They can be sorted in Month order or ascending or descending.

$cat sortsample.txt (display sample file contents)

$sort sortsample.txt (sorts all)

$sort +1 -2 -M sortsample.txt (sorts field 2 by month)



Part IIa (5 points each): 25pts

5. What is the output of the following sequence of bash commands: **echo 'Hello World' | sed 's/$/!!!/g'**

   The output appends !!! to the end of Hello World so it appears like:

   Hello World!!!

```
[jrogers75@gsuad.gsu.edu@snowball Homeworks]$ echo 'Hello World' | sed 's/$/!!!
/g'
Hello World!!!
```

6. What is the output for each of these awk script commands?

   -- 1 <= NF { print $5 }
   output: print the 5th field if the current line is less than or equal to 1

   -- NR >= 1 && NR >= 5 { print $1 }
   output: print the value in the first field of the line if the current line # is greater than or equal to 1 AND the current line number is greater than or equal to 5

   -- 1,5 { print $0 }
   output: print entire lines 1 through 5

   -- {print $1 }
   output: print the first field of each line

7. What is the output of following command line:
   echo good | sed '/Good/d'
   The output prints Good (replaces good with Good):

```
[jrogers75@gsuad.gsu.edu@snowball Homeworks]$ echo good | sed '/Good/d'
good
```

8. Which **awk** script outputs all the lines where a plus sign + appears at the end of line?

awk '/\+$/{print}' filename

Example:

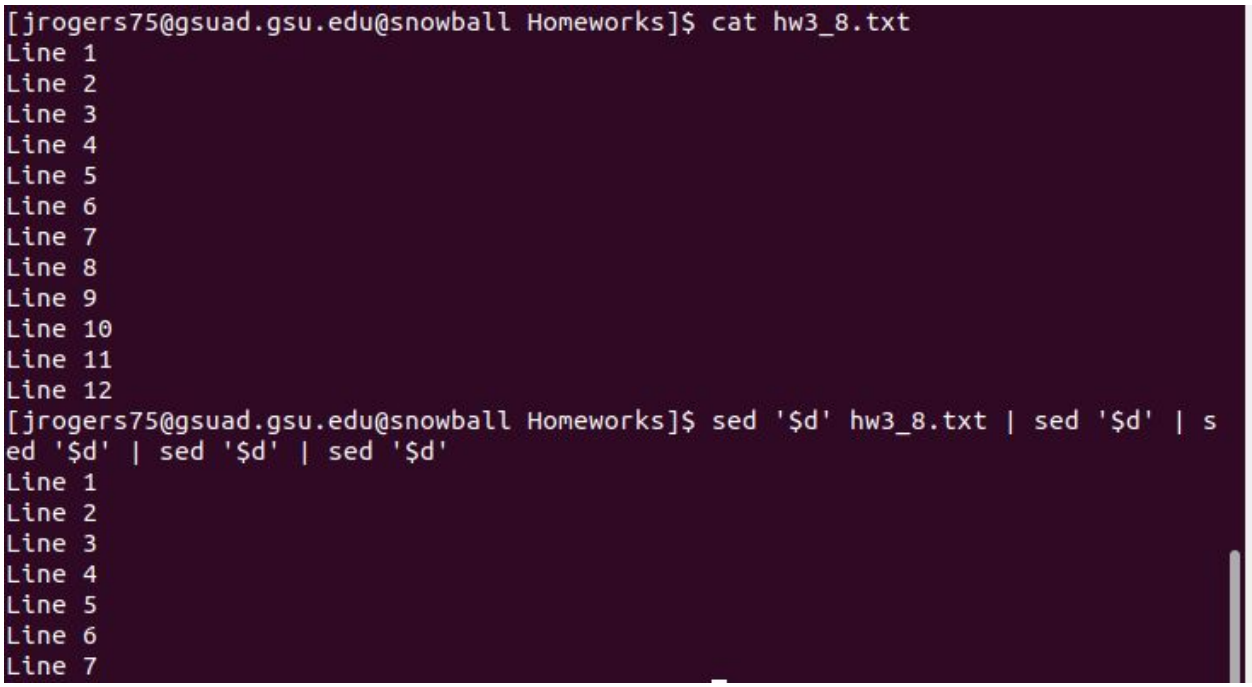
```
[jrogers75@gsuad.gsu.edu@snowball Homeworks]$ cat awktest8.txt
Need more text
Will this work?+
Hope I make a good grade +
I can't think of anything else to write.
End of the story. Yahoo+
[jrogers75@gsuad.gsu.edu@snowball Homeworks]$ awk '/\+$/{print}' awktest8.txt
Will this work?+
Hope I make a good grade +
End of the story. Yahoo+
```

9. What is the command to delete only the first 5 lines in a file "foo"? Which command deletes only the last 5 lines?

To delete first 5 lines: sed 1,5d filename

```
[jrogers75@gsuad.gsu.edu@snowball Homeworks]$ cat hw3_8.txt
Line 1
Line 2
Line 3
Line 4
Line 5
Line 6
Line 7
Line 8
Line 9
Line 10
Line 11
Line 12
[jrogers75@gsuad.gsu.edu@snowball Homeworks]$ sed 1,5d hw3_8.txt
Line 6
Line 7
Line 8
Line 9
Line 10
Line 11
Line 12
```

To delete last 5 lines: sed ‘$d’ filename | sed ‘$d’ | sed ‘$d’ | sed ‘$d’ | sed ‘$d’



Part IIb (10pts each): 50pts

Describe the function (5pts) and output (5pts) of the following commands.

9. **$ cat float** ***Displays contents of file named float**

```
[jrogers75@gsuad.gsu.edu@snowball Homeworks]$ cat float
Wish I was floating in blue across the sky, my imagination is strong,
And I often visit the days
When everything seemed so clear.
Now I wonder what I'm doing here at all...
```

$ cat h1.awk ***Displays contents of file named h1.awk

NR>2 && NR<4{print NR ":" $0

```
[jrogers75@gsuad.gsu.edu@snowball Homeworks]$ cat h1.awk
NR>2&&NR<4{print NR ":" $0}
```

***** The command in the file means to print the current line number followed by : and the first line number and line contents if the current line # is greater than 2 and less than 4**

$ awk '/.*ing/ {print NR ":" $1}' float

******If the line has a word with the string 'ing' in it, print the line number and the value in the first field.**

The output is: 1:Wish
3:When
4:Now

```
[jrogers75@gsuad.gsu.edu@snowball Homeworks]$ awk '/.*ing/ {print NR ":" $1}' f
loat
1:Wish
3:When
4:Now
```

10. As the next command following question 9,

$ awk -f h1.awk float

**** Means to print the entire line of the file that has a line number of more than 2 and less than 4.**

Output: 3:When everything seemed so clear.

```
[jrogers75@gsuad.gsu.edu@snowball Homeworks]$ awk -f h1.awk float
3:When everything seemed so clear.
```

11.

$ **cat h2.awk**

```
BEGIN { print    "Start to scan file" }
{print  $1      ","  $NF}
                  "END-" , FILENAME }
END {print
```

$ awk -f h2.awk float

$

Script function: print the text 'Start to scan file' when script is started. Print the value in the first field and last field of each line, then print END- and the filename.

******** Run the script in h2.awk against the file named float. This will give the following output:**

Start to scan file
Wish,strong,
And,days,
When,clear.
Now,all...
END- float

```
[jrogers75@gsuad.gsu.edu@snowball Homeworks]$ awk -f h2.awk float
Start to scan file
Wish,strong,
And,days
When,clear.
Now,all...
END- float
```

12. sed 's/\s/\t/g' float

This command is to change all the white space (spaces between words) into tabs so it makes the gaps between words larger.

The resulting output is something like this:

Wish I was floating in blue across the sky,
my imagination is strong, And I often
visit the days
When everything seemed so clear.
Now I wonder what I'm doing here at all…

```
[jrogers75@gsuad.gsu.edu@snowball Homeworks]$ sed 's/\s/\t/g' float
Wish    I       was     floating        in      blue    across  the     sky,    m
y       imagination     is      strong,
And     I       often   visit   the     days
When    everything      seemed  so      clear.
Now     I       wonder  what    I'm     doing   here    at      all...
```

13.

$ ls *.awk| awk '{print "grep --color 'BEGIN' " $1 }' |sh *(Notes: **sh file** runs file as a shell script . $1 should be the output of* ‘ ls *.awk ‘ in this case, not the 1st field)*

T In the command, ls *.awk is looking at the output of h1.awk and h2.awk and piping it two the second part of the command: awk ‘{print “grep -- color ‘BEGIN’ “$1 }’ and running it as a shell script. It ultimately printed the first line of the file named h2.awk because the grep search matched.

```
[jrogers75@gsuad.gsu.edu@snowball Homeworks]$ ls *awk | awk '{print "grep --col
or 'BEGIN' " $1}' | sh
BEGIN { print   "Start to scan file"}
```

14.

$ mkdir test test/test1 test/test2

**Creates a directory called test with subdirectories called test1 and test 2

$cat>test/testt.txt
This is a test file ^D

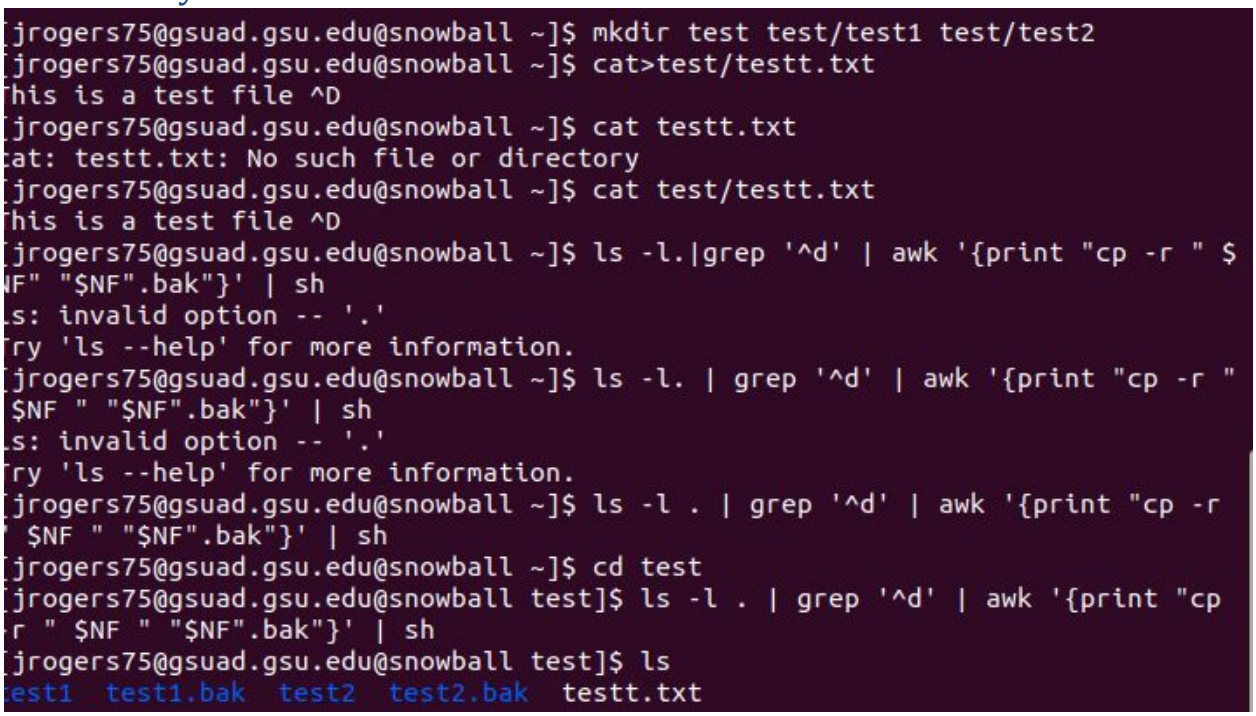
** Creates a file named testt.txt in directory called test. The text in the testt.txt file is "This is a test file ^D" . I know ^D is also Ctrl-D and it wasn't clear if this needed to be included in the testt.txt file or if it was an indicator to use Ctrl-D to save and quit the file.

$ cd test

** Change present working directory to the newly created 'test' directory

$ ls -l . | grep '^d' | awk '{print "cp -r " $NF " " $NF ".bak"}' | sh

** Created new subdirectories called test1.bak and test2.bak within the test directory.



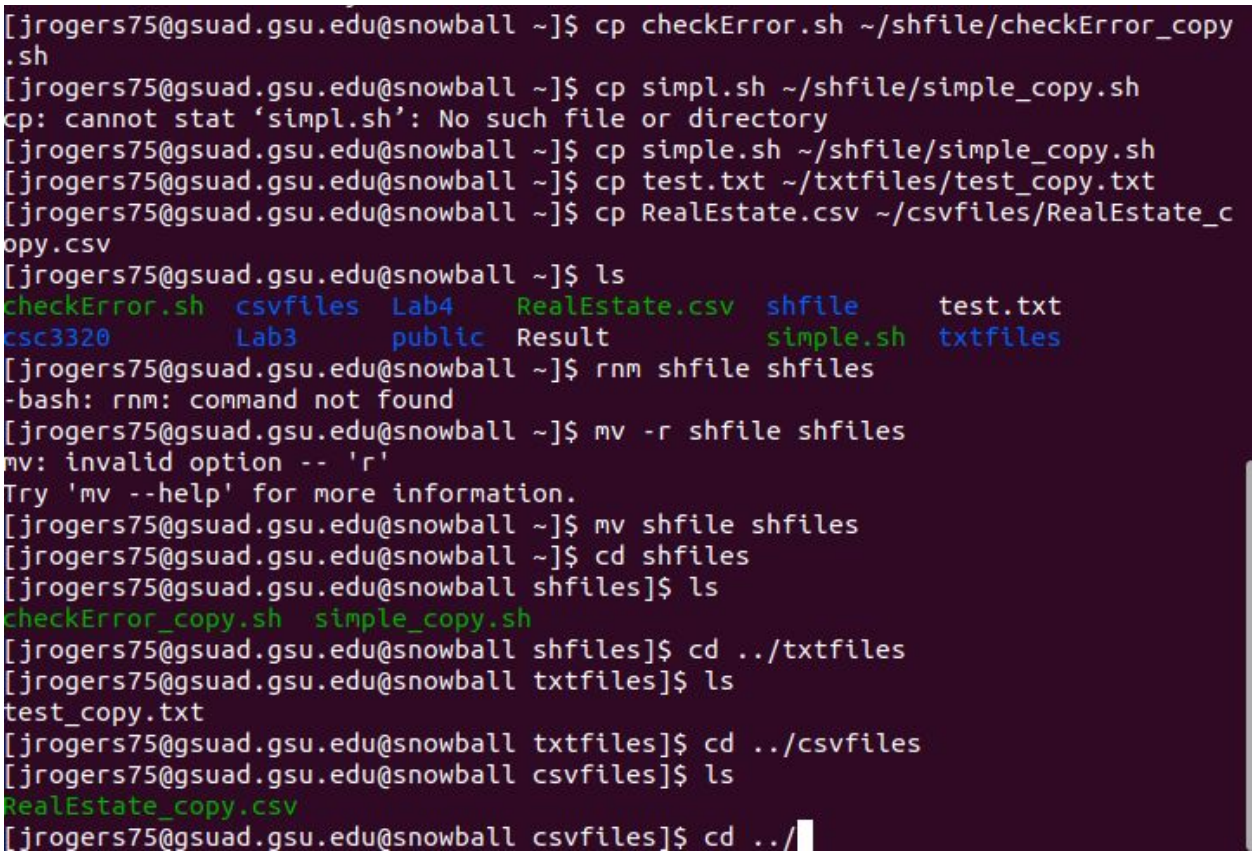
Part III Programming: 15pts

15. Sort all the files in your class working directory (or your home directory) as per the following requirements:

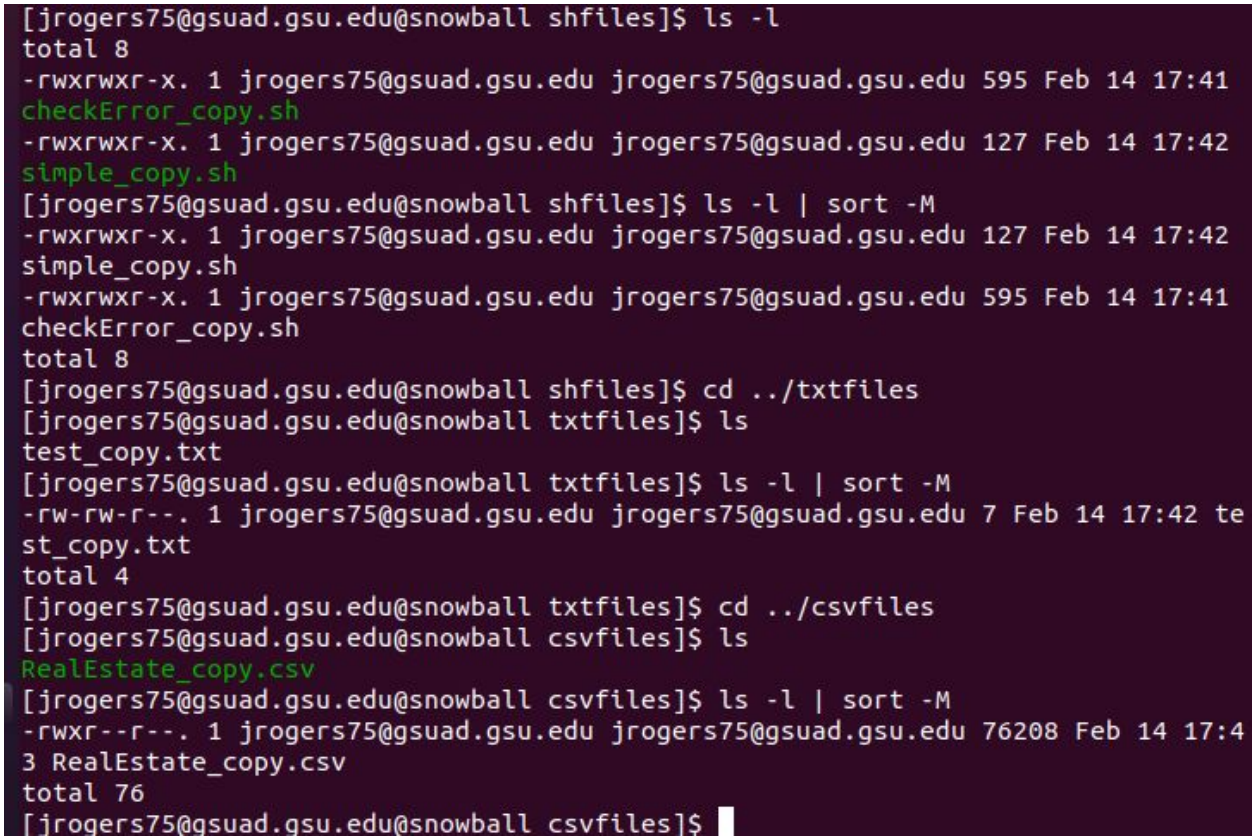
a. A copy of each file in that folder must be made. Append the string "_copy" to the name of the file

b. The duplicate (copied) files must be in separate directories with each directory specifying the type of the file (e.g. txt files in directory named txtfiles, pdf files in directory named pdffiles etc).

a&b steps in screenshot below:



c. The files in each directory must be sorted in chronological order of months. ** all the copied files were created the same day (2/14/21) so they all have the same date and sorting didn't make any changes

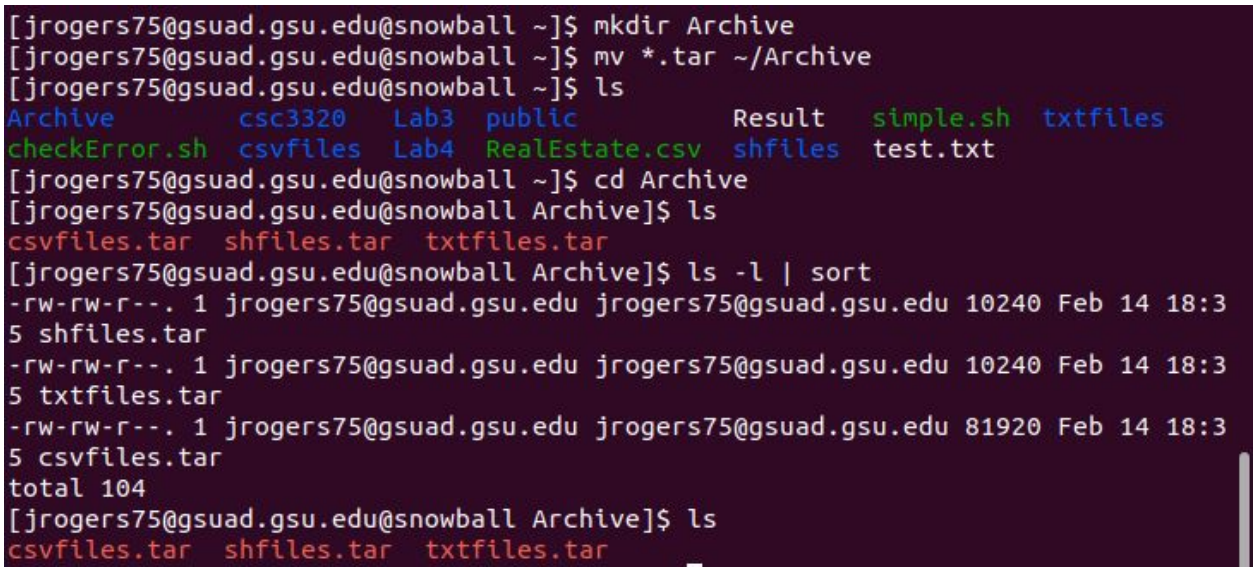


d. An archive file (.tar) of each directory must be made. The .tar files must be sorted by name in ascending order.

```
[jrogers75@gsuad.gsu.edu@snowball ~]$ tar cvf txtfiles.tar txtfiles
txtfiles/
txtfiles/test_copy.txt
[jrogers75@gsuad.gsu.edu@snowball ~]$ tar cvf csvfiles.tar csvfiles
csvfiles/
csvfiles/RealEstate_copy.csv
[jrogers75@gsuad.gsu.edu@snowball ~]$ tar cvf shfiles.tar shfiles
shfiles/
shfiles/simple_copy.sh
shfiles/checkError_copy.sh
[jrogers75@gsuad.gsu.edu@snowball ~]$ ls
checkError.sh  csvfiles.tar  public          shfiles      test.txt
csc3320        Lab3          RealEstate.csv  shfiles.tar  txtfiles
csvfiles       Lab4          Result          simple.sh    txtfiles.tar
[jrogers75@gsuad.gsu.edu@snowball ~]$ ls -l | sort
drwx------. 2 jrogers75@gsuad.gsu.edu jrogers75@gsuad.gsu.edu  4096 Feb  3 15:0
6 Lab3
drwxrwxr-x. 2 jrogers75@gsuad.gsu.edu jrogers75@gsuad.gsu.edu  4096 Feb 11 19:0
2 Lab4
drwxrwxr-x. 2 jrogers75@gsuad.gsu.edu jrogers75@gsuad.gsu.edu  4096 Feb 14 17:4
2 shfiles
drwxrwxr-x. 2 jrogers75@gsuad.gsu.edu jrogers75@gsuad.gsu.edu  4096 Feb 14 17:4
2 txtfiles
drwxrwxr-x. 2 jrogers75@gsuad.gsu.edu jrogers75@gsuad.gsu.edu  4096 Feb 14 17:4
3 csvfiles
drwxrwxr-x. 3 jrogers75@gsuad.gsu.edu jrogers75@gsuad.gsu.edu  4096 Jan 21 18:5
1 csc3320
drwxrwxr-x. 5 jrogers75@gsuad.gsu.edu jrogers75@gsuad.gsu.edu  4096 Jan 28 10:3
```

```
1 csc3320
drwxrwxr-x. 5 jrogers75@gsuad.gsu.edu jrogers75@gsuad.gsu.edu  4096 Jan 28 10:3
0 public
-rw-rw-r--. 1 jrogers75@gsuad.gsu.edu jrogers75@gsuad.gsu.edu 10240 Feb 14 18:3
5 shfiles.tar
-rw-rw-r--. 1 jrogers75@gsuad.gsu.edu jrogers75@gsuad.gsu.edu 10240 Feb 14 18:3
5 txtfiles.tar
-rw-rw-r--. 1 jrogers75@gsuad.gsu.edu jrogers75@gsuad.gsu.edu    49 Feb 12 15:1
5 Result
-rw-rw-r--. 1 jrogers75@gsuad.gsu.edu jrogers75@gsuad.gsu.edu     7 Feb  3 15:0
6 test.txt
-rw-rw-r--. 1 jrogers75@gsuad.gsu.edu jrogers75@gsuad.gsu.edu 81920 Feb 14 18:3
5 csvfiles.tar
-rwxr--r--. 1 jrogers75@gsuad.gsu.edu jrogers75@gsuad.gsu.edu 76208 Feb  1 20:5
9 RealEstate.csv
-rwxrwxr-x. 1 jrogers75@gsuad.gsu.edu jrogers75@gsuad.gsu.edu   127 Feb 12 14:3
7 simple.sh
-rwxrwxr-x. 1 jrogers75@gsuad.gsu.edu jrogers75@gsuad.gsu.edu   595 Feb 12 15:1
4 checkError.sh
total 224
```

e. An archive file of all the .tar archive files must be made and be available in your home directory.



As an output, show your screen shots for each step or a single screenshot that will cover the outputs from all the steps.