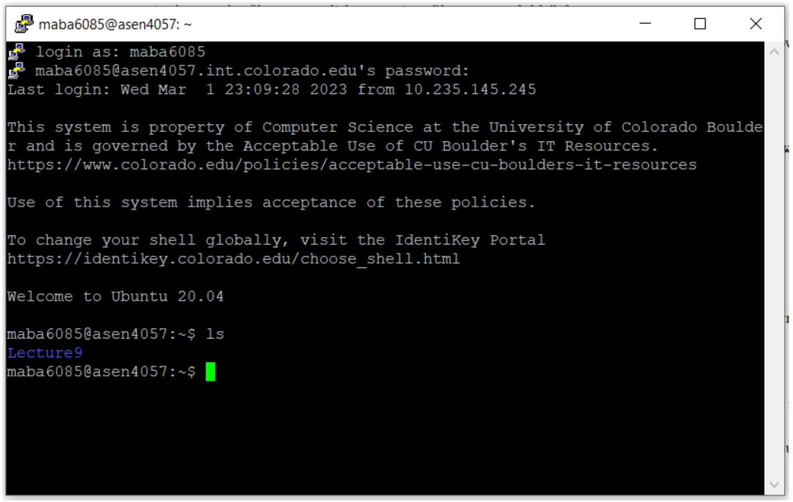
Elena Bauer
Professor Le Moine
ASEN 4057

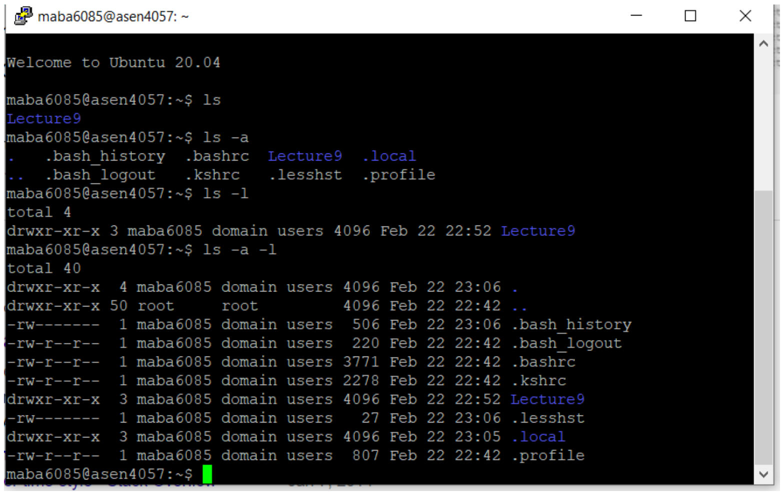
Homework #3

1. Log into your CSCI OpenStack account. Enter the command to display the directory contents



```
maba6085@asen4057: ~
 login as: maba6085
 maba6085@asen4057.int.colorado.edu's password:
Last login: Wed Mar  1 23:09:28 2023 from 10.235.145.245

This system is property of Computer Science at the University of Colorado Boulde
r and is governed by the Acceptable Use of CU Boulder's IT Resources.
https://www.colorado.edu/policies/acceptable-use-cu-boulders-it-resources

Use of this system implies acceptance of these policies.

To change your shell globally, visit the IdentiKey Portal
https://identikey.colorado.edu/choose_shell.html

Welcome to Ubuntu 20.04

maba6085@asen4057:~$ ls
Lecture9
maba6085@asen4057:~$
```

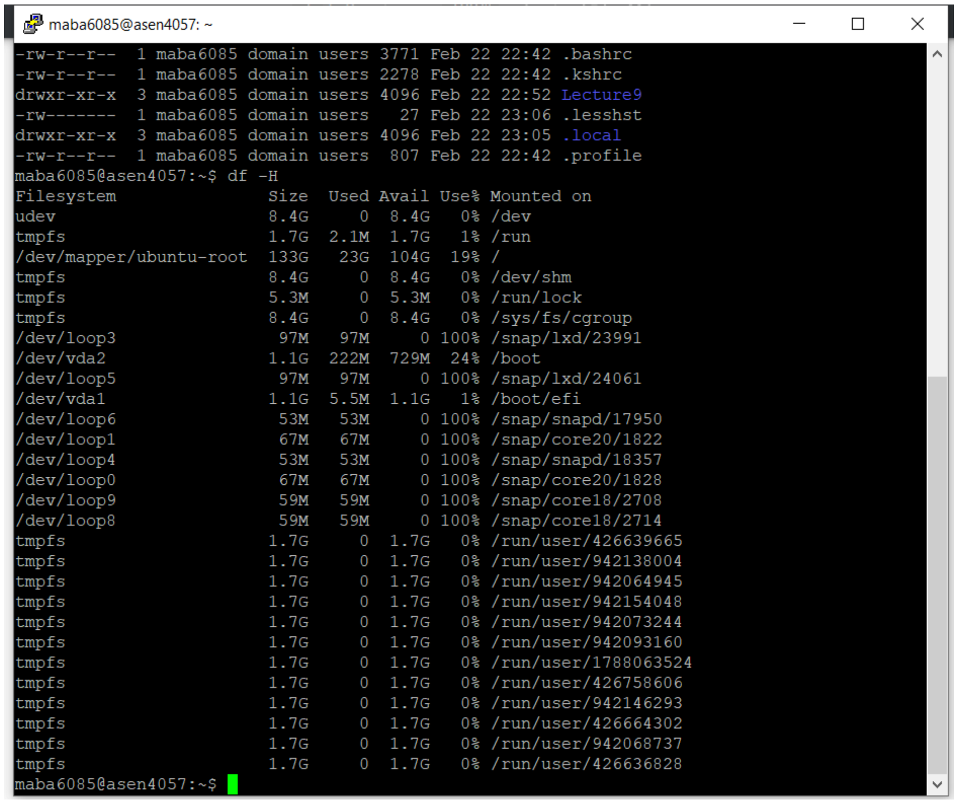
2. Get the directory listing showing all files (including hidden files) within the directory in a ”long” format



```
maba6085@asen4057: ~

Welcome to Ubuntu 20.04

maba6085@asen4057:~$ ls
Lecture9
maba6085@asen4057:~$ ls -a
.    .bash_history  .bashrc  Lecture9  .local
..   .bash_logout   .kshrc   .lesshst  .profile
maba6085@asen4057:~$ ls -l
total 4
drwxr-xr-x 3 maba6085 domain users 4096 Feb 22 22:52 Lecture9
maba6085@asen4057:~$ ls -a -l
total 40
drwxr-xr-x  4 maba6085 domain users 4096 Feb 22 23:06 .
drwxr-xr-x 50 root     root         4096 Feb 22 22:42 ..
-rw-------  1 maba6085 domain users  506 Feb 22 23:06 .bash_history
-rw-r--r--  1 maba6085 domain users  220 Feb 22 22:42 .bash_logout
-rw-r--r--  1 maba6085 domain users 3771 Feb 22 22:42 .bashrc
-rw-r--r--  1 maba6085 domain users 2278 Feb 22 22:42 .kshrc
drwxr-xr-x  3 maba6085 domain users 4096 Feb 22 22:52 Lecture9
-rw-------  1 maba6085 domain users   27 Feb 22 23:06 .lesshst
drwxr-xr-x  3 maba6085 domain users 4096 Feb 22 23:05 .local
-rw-r--r--  1 maba6085 domain users  807 Feb 22 22:42 .profile
maba6085@asen4057:~$
```

3. For each file listed in the long format, what do each of the first ten characters imply?
   - The first character refers to the file type
     - For example in the code above, Lecture 9 has “d” which stands for directory
   - The rest of the characters refer to the permission settings
     - So for example for Lecture 9, domain, local, and root:
       - The first set of 3 characters “rwx” shows the user owner has read, write, and execute permissions
       - The second set of characters “r-x” shows the group owner has read and execute permissions

        - Lastly, the third set “r-x” shows that others have read and execute permissions
    - For .bash_history and .lesshst,
        - The user owner only has read and write permissions. The group and others do not have permissions
    - For .bash_logout, .bashrc, .kshrc, and .profile
        - The user owner has read and write permissions
        - The group owner has read permissions
        - Others have read permissions
4. Access the file system disk usage in a ”human readable” format



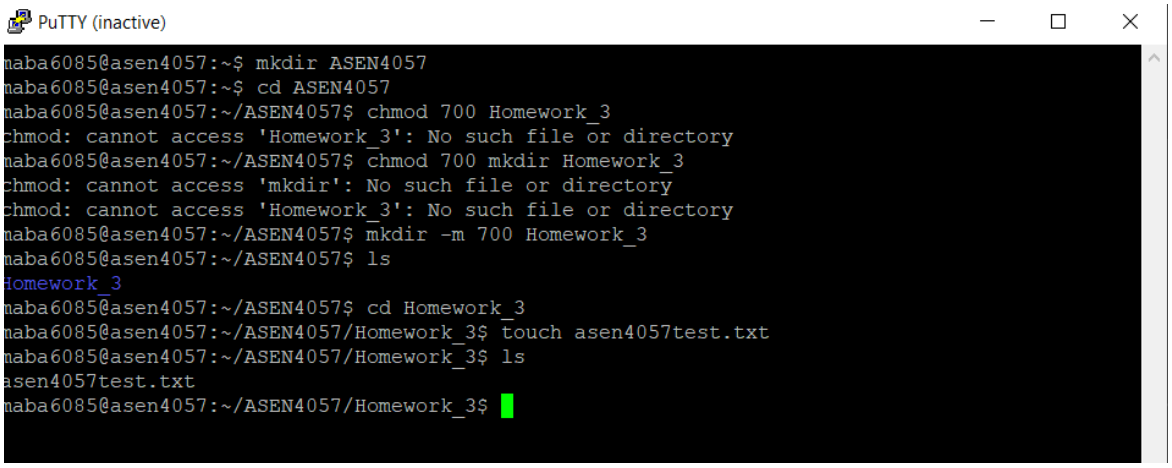
maba6085@asen4057: ~

```
-rw-r--r--  1 maba6085 domain users 3771 Feb 22 22:42 .bashrc
-rw-r--r--  1 maba6085 domain users 2278 Feb 22 22:42 .kshrc
drwxr-xr-x  3 maba6085 domain users 4096 Feb 22 22:52 Lecture9
-rw-------  1 maba6085 domain users   27 Feb 22 23:06 .lesshst
drwxr-xr-x  3 maba6085 domain users 4096 Feb 22 23:05 .local
-rw-r--r--  1 maba6085 domain users  807 Feb 22 22:42 .profile
maba6085@asen4057:~$ df -H
Filesystem                 Size  Used Avail Use% Mounted on
udev                       8.4G     0  8.4G   0% /dev
tmpfs                      1.7G  2.1M  1.7G   1% /run
/dev/mapper/ubuntu-root    133G   23G  104G  19% /
tmpfs                      8.4G     0  8.4G   0% /dev/shm
tmpfs                      5.3M     0  5.3M   0% /run/lock
tmpfs                      8.4G     0  8.4G   0% /sys/fs/cgroup
/dev/loop3                  97M   97M     0 100% /snap/lxd/23991
/dev/vda2                  1.1G  222M  729M  24% /boot
/dev/loop5                  97M   97M     0 100% /snap/lxd/24061
/dev/vda1                  1.1G  5.5M  1.1G   1% /boot/efi
/dev/loop6                  53M   53M     0 100% /snap/snapd/17950
/dev/loop1                  67M   67M     0 100% /snap/core20/1822
/dev/loop4                  53M   53M     0 100% /snap/snapd/18357
/dev/loop0                  67M   67M     0 100% /snap/core20/1828
/dev/loop9                  59M   59M     0 100% /snap/core18/2708
/dev/loop8                  59M   59M     0 100% /snap/core18/2714
tmpfs                      1.7G     0  1.7G   0% /run/user/426639665
tmpfs                      1.7G     0  1.7G   0% /run/user/942138004
tmpfs                      1.7G     0  1.7G   0% /run/user/942064945
tmpfs                      1.7G     0  1.7G   0% /run/user/942154048
tmpfs                      1.7G     0  1.7G   0% /run/user/942073244
tmpfs                      1.7G     0  1.7G   0% /run/user/942093160
tmpfs                      1.7G     0  1.7G   0% /run/user/1788063524
tmpfs                      1.7G     0  1.7G   0% /run/user/426758606
tmpfs                      1.7G     0  1.7G   0% /run/user/942146293
tmpfs                      1.7G     0  1.7G   0% /run/user/426664302
tmpfs                      1.7G     0  1.7G   0% /run/user/942068737
tmpfs                      1.7G     0  1.7G   0% /run/user/426636828
maba6085@asen4057:~$
```

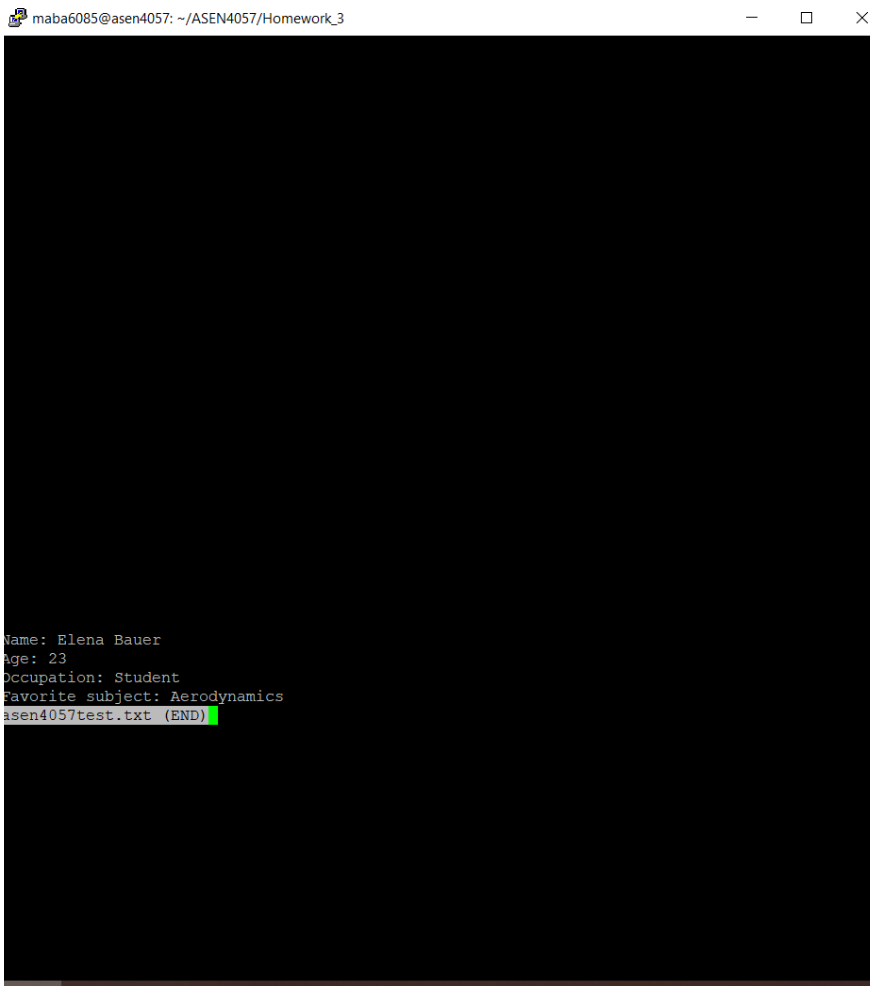
5. Examine the contents of the /bin directory. Choose 5 programs that you do not know and determine what they do and how they are called using the command line.
    - **nroff**: nroff is a command that stands for “new runoff” and it is used to format documents for display
        - It is called by: nroff [options] [filename]
    - **curl**: curl is a command to transfer data from or to a server
        - It is called by: curl [options] [URL]
    - **sleep**: sleep is a command that is used to pause or delay for an amount of time
        - It is called by: sleep NUMBER[SUFFIX]
            - Where suffix can be s, for seconds, m, for minutes, h, for hours, and d, for days.
    - **mcookie:** mcookie is a command that generates a 128-bit random hex number for use in the X authority system.

  - It is called by: mcookie [options] [filename]
- **join**: join is a command that joins lines of two files onto a common field
  - It is called by: join [options] file1 file2

6. Complete the following tasks to create new directories and files:
   a. Create a folder in your home directory called ASEN4057
   b. In one command, create a folder in ASEN4057 called Homework_3 that gives read, write, and execute permissions to the user only
   c. Within this new directory, create a text file called asen4057test.txt
   d. Without opening asen4057test.txt, add the following information:
   Name: [Enter your first and last name here]
   Age: [Enter your age here]
   Occupation: [Enter your occupation here]
   Favorite subject: [Enter your favorite subject here]
   e. Without opening the file, display the entire content of asen4057test.txt to the terminal


PuTTY (inactive)

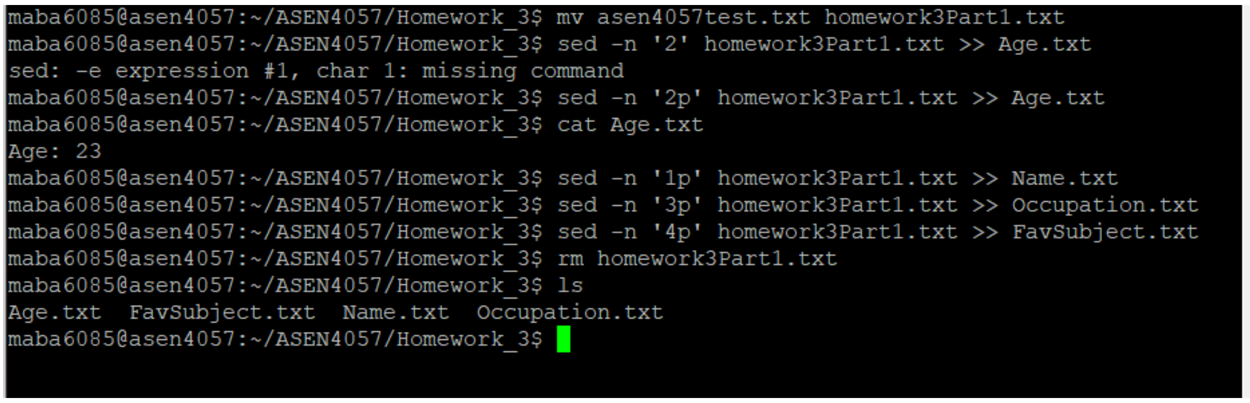
```
naba6085@asen4057:~$ mkdir ASEN4057
naba6085@asen4057:~$ cd ASEN4057
naba6085@asen4057:~/ASEN4057$ chmod 700 Homework_3
chmod: cannot access 'Homework_3': No such file or directory
naba6085@asen4057:~/ASEN4057$ chmod 700 mkdir Homework_3
chmod: cannot access 'mkdir': No such file or directory
chmod: cannot access 'Homework_3': No such file or directory
naba6085@asen4057:~/ASEN4057$ mkdir -m 700 Homework_3
naba6085@asen4057:~/ASEN4057$ ls
Homework_3
naba6085@asen4057:~/ASEN4057$ cd Homework_3
naba6085@asen4057:~/ASEN4057/Homework_3$ touch asen4057test.txt
naba6085@asen4057:~/ASEN4057/Homework_3$ ls
asen4057test.txt
naba6085@asen4057:~/ASEN4057/Homework_3$
```



```
maba6085@asen4057: ~/ASEN4057/Homework_3
```

```
Name: Elena Bauer
Age: 23
Occupation: Student
Favorite subject: Aerodynamics
asen4057test.txt (END)
```

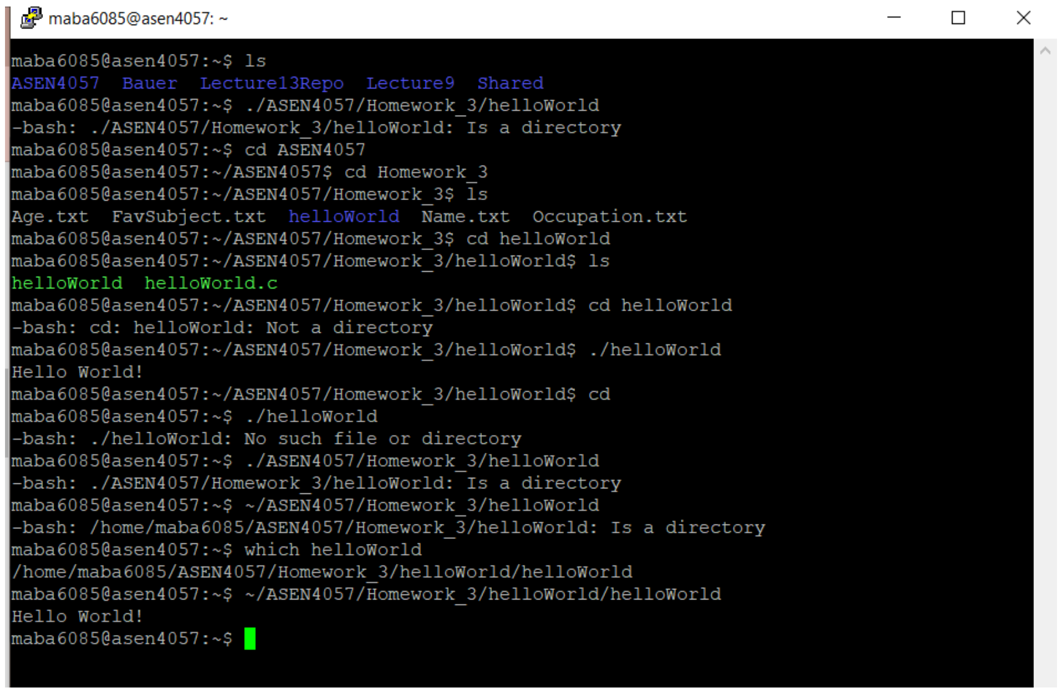
7. Complete the following tasks to execute multiple commands in one line:
    a. Rename asen4057test.txt to homework3Part1.txt
    b. Without opening homework3Part1.txt, take the second line in the file and copy it into a new file called Age.txt.
    c. Perform the same tasks on line 1, 3, and 4 of homework3Part1.txt and store the results in Name.txt, Occupation.txt, and FavSubject.txt respectively.
    d. Delete homework3Part1.txt and verify that is has been deleted



```
maba6085@asen4057:~/ASEN4057/Homework_3$ mv asen4057test.txt homework3Part1.txt
maba6085@asen4057:~/ASEN4057/Homework_3$ sed -n '2' homework3Part1.txt >> Age.txt
sed: -e expression #1, char 1: missing command
maba6085@asen4057:~/ASEN4057/Homework_3$ sed -n '2p' homework3Part1.txt >> Age.txt
maba6085@asen4057:~/ASEN4057/Homework_3$ cat Age.txt
Age: 23
maba6085@asen4057:~/ASEN4057/Homework_3$ sed -n '1p' homework3Part1.txt >> Name.txt
maba6085@asen4057:~/ASEN4057/Homework_3$ sed -n '3p' homework3Part1.txt >> Occupation.txt
maba6085@asen4057:~/ASEN4057/Homework_3$ sed -n '4p' homework3Part1.txt >> FavSubject.txt
maba6085@asen4057:~/ASEN4057/Homework_3$ rm homework3Part1.txt
maba6085@asen4057:~/ASEN4057/Homework_3$ ls
Age.txt  FavSubject.txt  Name.txt  Occupation.txt
maba6085@asen4057:~/ASEN4057/Homework_3$
```

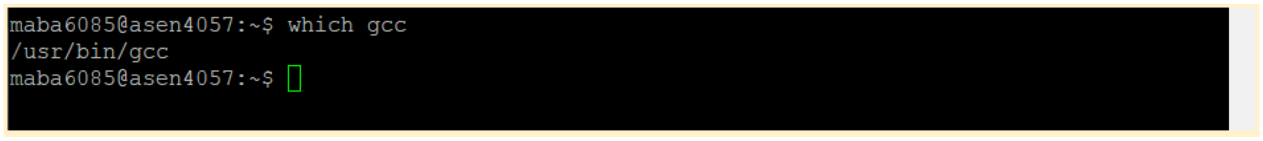
8. Complete the following tasks to create a C program called helloWorld:

a. Create a new directory in the Homework_3 directory called helloWorld
b. Within this new directory, create a new text tile called helloWorld.c which contains the following:
```
/*File: helloWorld.c*/
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char** argv)
{
printf("Hello World!\n");
return (EXIT_SUCCESS);
}
```
c. Compile the code and create an executable file called helloWorld using the following gcc command
```
gcc helloWorld.c -o helloWorld
```
d. Verify that you have execute permissions on the helloWorld executable.
e. Run the code (./helloWorld) and verify the output in the terminal
f. Move to your home directory and run the helloWorld program from your home directory



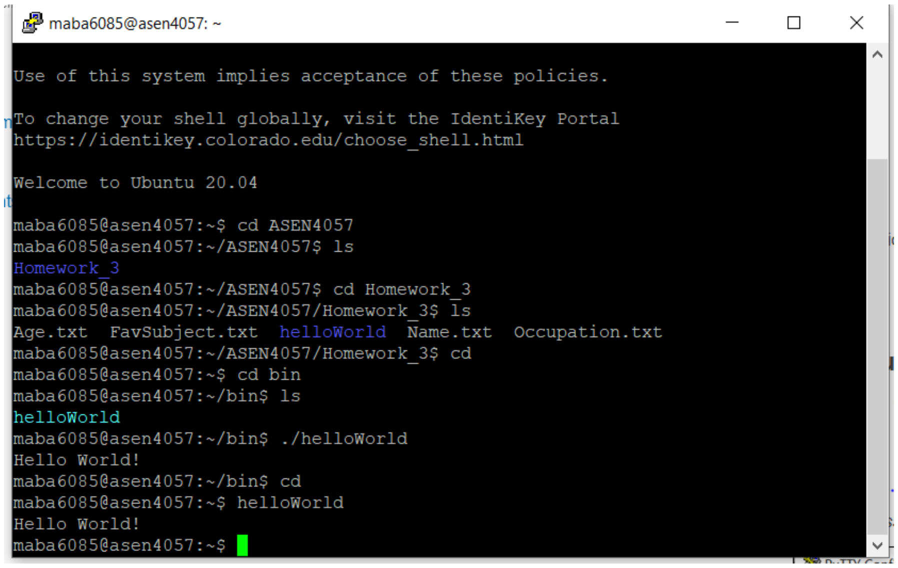
```
maba6085@asen4057:~$ ls
ASEN4057  Bauer  Lecture13Repo  Lecture9  Shared
maba6085@asen4057:~$ ./ASEN4057/Homework_3/helloWorld
-bash: ./ASEN4057/Homework_3/helloWorld: Is a directory
maba6085@asen4057:~$ cd ASEN4057
maba6085@asen4057:~/ASEN4057$ cd Homework_3
maba6085@asen4057:~/ASEN4057/Homework_3$ ls
Age.txt  FavSubject.txt  helloWorld  Name.txt  Occupation.txt
maba6085@asen4057:~/ASEN4057/Homework_3$ cd helloWorld
maba6085@asen4057:~/ASEN4057/Homework_3/helloWorld$ ls
helloWorld  helloWorld.c
maba6085@asen4057:~/ASEN4057/Homework_3/helloWorld$ cd helloWorld
-bash: cd: helloWorld: Not a directory
maba6085@asen4057:~/ASEN4057/Homework_3/helloWorld$ ./helloWorld
Hello World!
maba6085@asen4057:~/ASEN4057/Homework_3/helloWorld$ cd
maba6085@asen4057:~$ ./helloWorld
-bash: ./helloWorld: No such file or directory
maba6085@asen4057:~$ ./ASEN4057/Homework_3/helloWorld
-bash: ./ASEN4057/Homework_3/helloWorld: Is a directory
maba6085@asen4057:~$ ~/ASEN4057/Homework_3/helloWorld
-bash: /home/maba6085/ASEN4057/Homework_3/helloWorld: Is a directory
maba6085@asen4057:~$ which helloWorld
/home/maba6085/ASEN4057/Homework_3/helloWorld/helloWorld
maba6085@asen4057:~$ ~/ASEN4057/Homework_3/helloWorld/helloWorld
Hello World!
maba6085@asen4057:~$
```

9. Search the filestore for the location of the gcc executable. Why can you use gcc from the command line without providing the full path to its executable?
   - You can use gcc from the command line without providing the full path to its executable because it is provided universally from the PATH since /usr/bin is contained in the PATH



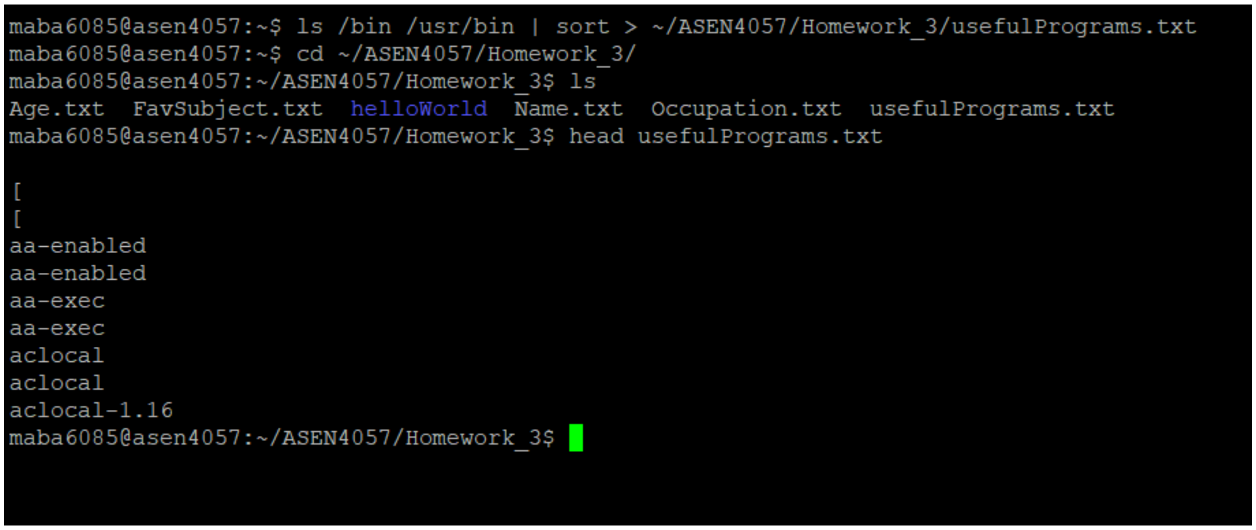
```
maba6085@asen4057:~$ which gcc
/usr/bin/gcc
maba6085@asen4057:~$
```

10. Complete the following tasks to set up a /bin folder in your home directory:
    a. Create a directory called /bin in your home directory
    b. Create a soft link to the helloWorld executable in your home /bin
    c. Add your home /bin do your PATH
    d. Test running your helloWorld executable from a any directory using the ./helloWorld command only (i.e., do not specify the path)



```
Use of this system implies acceptance of these policies.

To change your shell globally, visit the IdentiKey Portal
https://identikey.colorado.edu/choose_shell.html

Welcome to Ubuntu 20.04

maba6085@asen4057:~$ cd ASEN4057
maba6085@asen4057:~/ASEN4057$ ls
Homework_3
maba6085@asen4057:~/ASEN4057$ cd Homework_3
maba6085@asen4057:~/ASEN4057/Homework_3$ ls
Age.txt  FavSubject.txt  helloWorld  Name.txt  Occupation.txt
maba6085@asen4057:~/ASEN4057/Homework_3$ cd
maba6085@asen4057:~$ cd bin
maba6085@asen4057:~/bin$ ls
helloWorld
maba6085@asen4057:~/bin$ ./helloWorld
Hello World!
maba6085@asen4057:~/bin$ cd
maba6085@asen4057:~$ helloWorld
Hello World!
maba6085@asen4057:~$
```

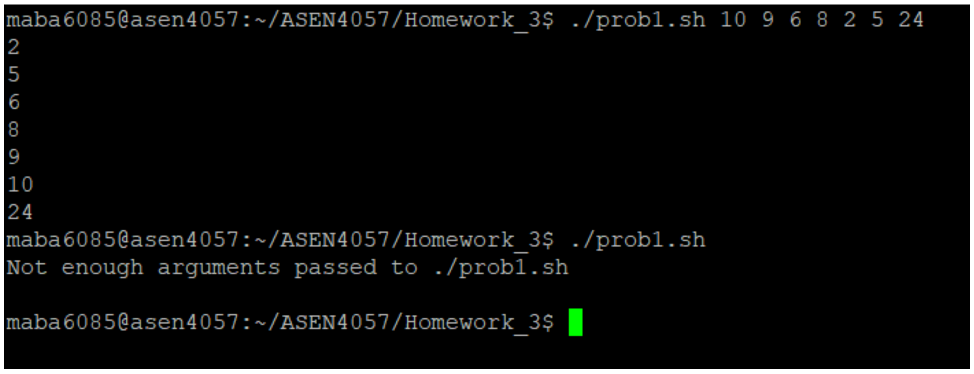
11. Using a single command line instruction, take the combined contents of /bin and /usr/bin, sort them, and write them to a file in the Homework_3 directory called usefulPrograms.txt



```
maba6085@asen4057:~$ ls /bin /usr/bin | sort > ~/ASEN4057/Homework_3/usefulPrograms.txt
maba6085@asen4057:~$ cd ~/ASEN4057/Homework_3/
maba6085@asen4057:~/ASEN4057/Homework_3$ ls
Age.txt  FavSubject.txt  helloWorld  Name.txt  Occupation.txt  usefulPrograms.txt
maba6085@asen4057:~/ASEN4057/Homework_3$ head usefulPrograms.txt

[
[
aa-enabled
aa-enabled
aa-exec
aa-exec
aclocal
aclocal
aclocal-1.16
maba6085@asen4057:~/ASEN4057/Homework_3$
```

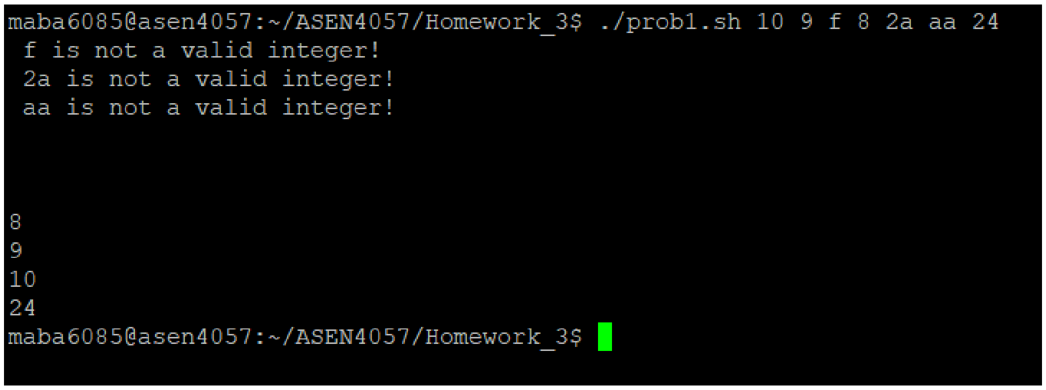
Problem 1: Sorting

Write a Bash shell script called prob1.sh which takes in n number of inputs, sorts them, and displays the sorted list. Note that the inputs should be a list of integer values. Additionally, your script should display an error message if the user provides no input.



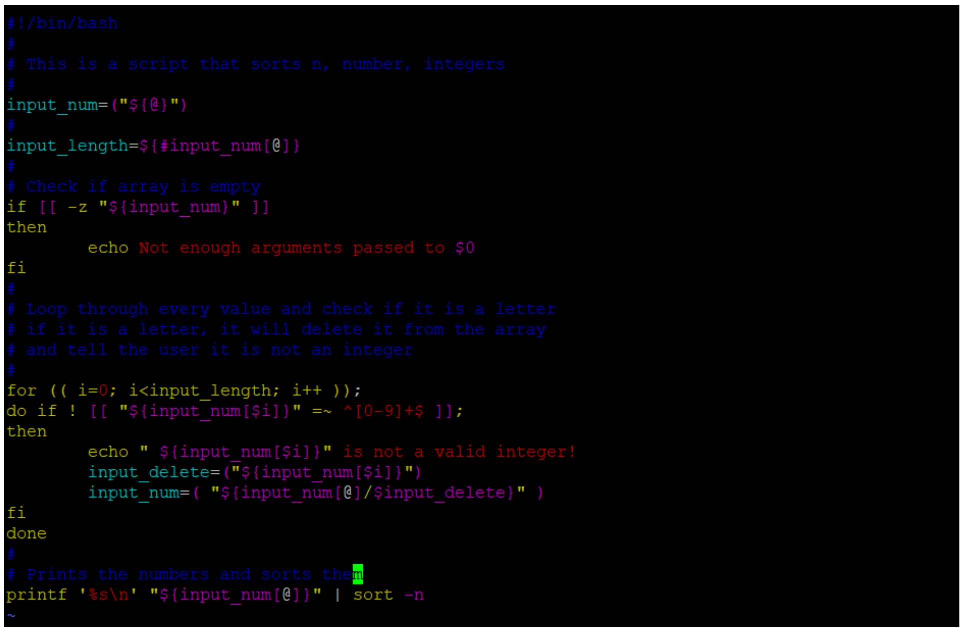
```
maba6085@asen4057:~/ASEN4057/Homework_3$ ./prob1.sh 10 9 6 8 2 5 24
2
5
6
8
9
10
24
maba6085@asen4057:~/ASEN4057/Homework_3$ ./prob1.sh
Not enough arguments passed to ./prob1.sh

maba6085@asen4057:~/ASEN4057/Homework_3$ 
```

Extra Credit:



```
maba6085@asen4057:~/ASEN4057/Homework_3$ ./prob1.sh 10 9 f 8 2a aa 24
 f is not a valid integer!
 2a is not a valid integer!
 aa is not a valid integer!




8
9
10
24
maba6085@asen4057:~/ASEN4057/Homework_3$ 
```

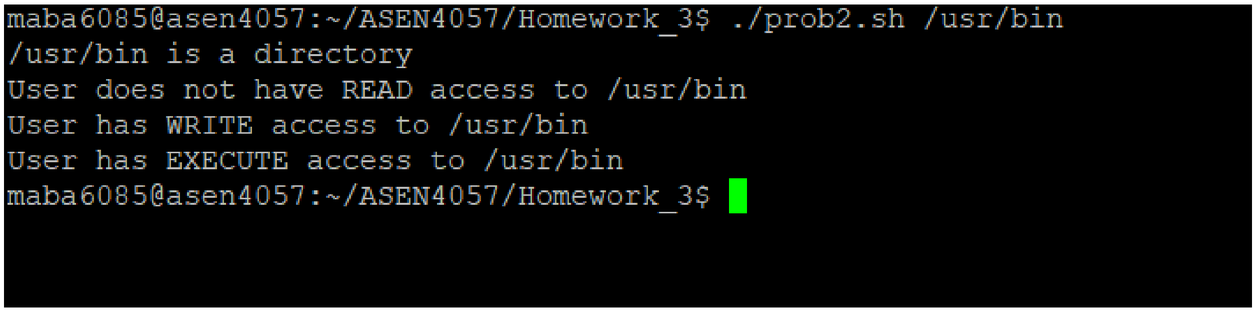
Script:



```
#!/bin/bash
#
# This is a script that sorts n, number, integers
#
input_num=("${@}")
#
input_length=${#input_num[@]}
#
# Check if array is empty
if [[ -z "${input_num}" ]]
then
        echo Not enough arguments passed to $0
fi
#
# Loop through every value and check if it is a letter
# if it is a letter, it will delete it from the array
# and tell the user it is not an integer
#
for (( i=0; i<input_length; i++ ));
do if ! [[ "${input_num[$i]}" =~ ^[0-9]+$ ]];
then
        echo " ${input_num[$i]}" is not a valid integer!
        input_delete=("${input_num[$i]}")
        input_num=( "${input_num[@]/$input_delete}" )
fi
done
#
# Prints the numbers and sorts them
printf '%s\n' "${input_num[@]}" | sort -n
~
```

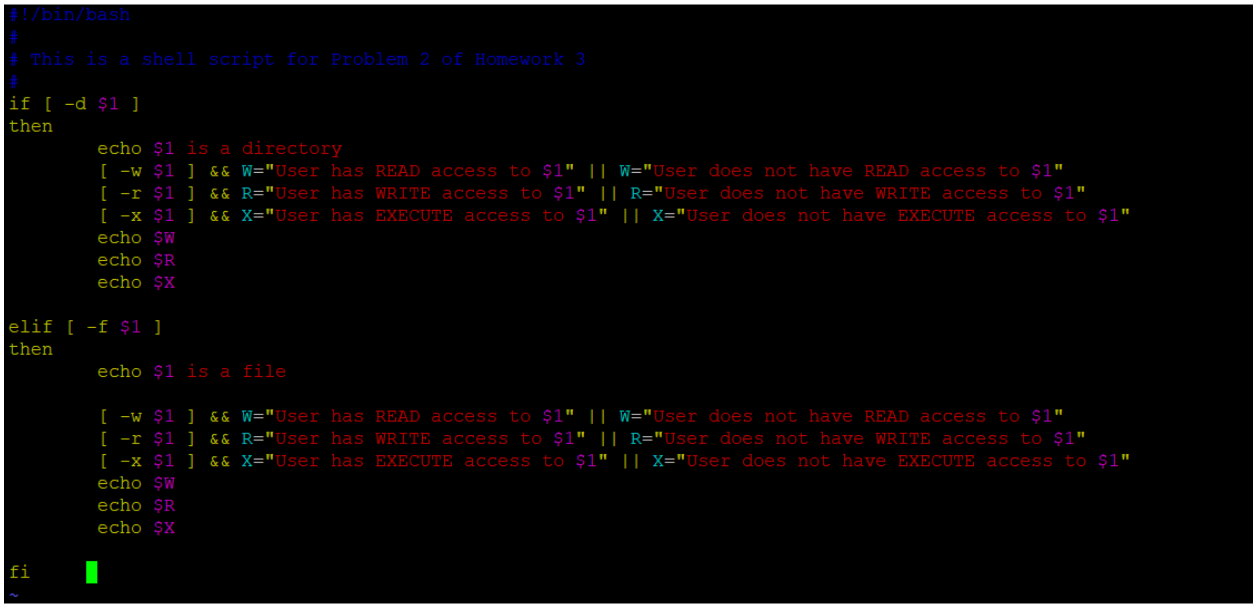
Problem 2: File or Directory

Write a Bash shell script called prob2.sh that takes the name of a file or directory as an input argument and reports if the file is a directory, regular file or other. Also report if the user has read, write, and execute permissions on the file or directory. There are several ways to approach this problem, but you may find it useful to use some combination of ls, read, head, and tail. You will want to use special options with these commands (i.e., special parameters).



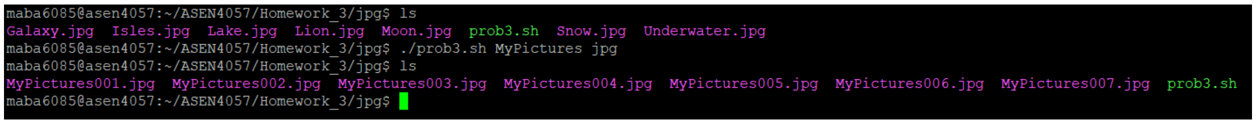
```
maba6085@asen4057:~/ASEN4057/Homework_3$ ./prob2.sh /usr/bin
/usr/bin is a directory
User does not have READ access to /usr/bin
User has WRITE access to /usr/bin
User has EXECUTE access to /usr/bin
maba6085@asen4057:~/ASEN4057/Homework_3$
```

Script:



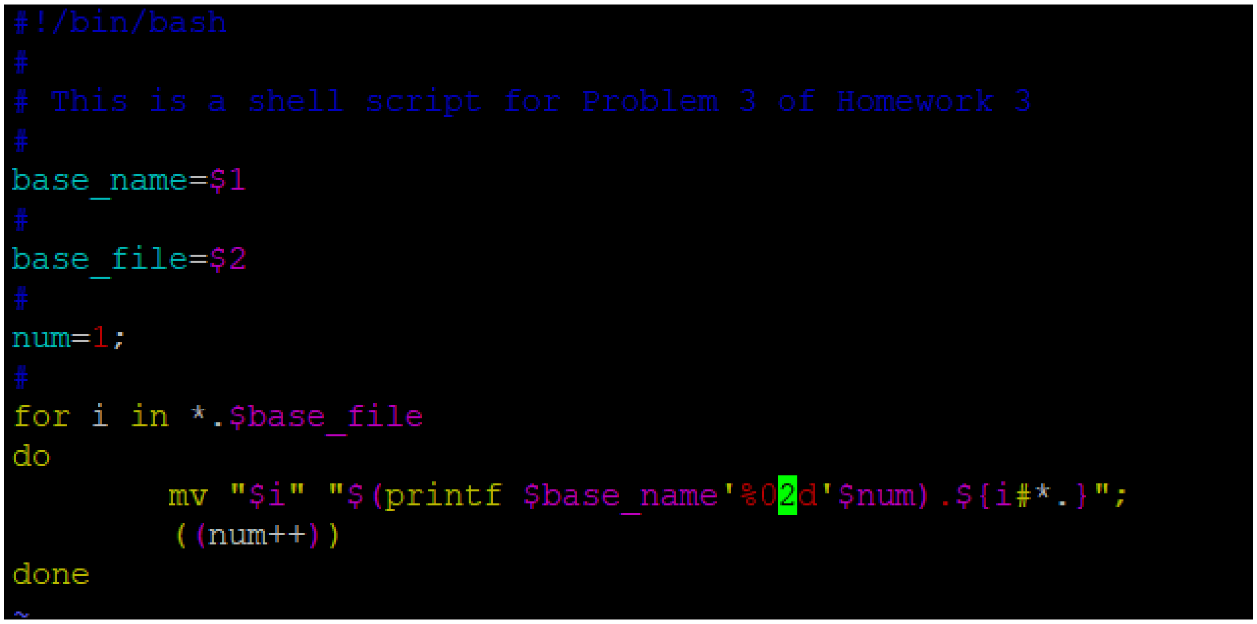
```
#!/bin/bash
#
# This is a shell script for Problem 2 of Homework 3
#
if [ -d $1 ]
then
        echo $1 is a directory
        [ -w $1 ] && W="User has READ access to $1" || W="User does not have READ access to $1"
        [ -r $1 ] && R="User has WRITE access to $1" || R="User does not have WRITE access to $1"
        [ -x $1 ] && X="User has EXECUTE access to $1" || X="User does not have EXECUTE access to $1"
        echo $W
        echo $R
        echo $X

elif [ -f $1 ]
then
        echo $1 is a file

        [ -w $1 ] && W="User has READ access to $1" || W="User does not have READ access to $1"
        [ -r $1 ] && R="User has WRITE access to $1" || R="User does not have WRITE access to $1"
        [ -x $1 ] && X="User has EXECUTE access to $1" || X="User does not have EXECUTE access to $1"
        echo $W
        echo $R
        echo $X

fi
~
```

Problem 3: Batch File Rename
Write a Bash shell script called prob3.sh that will rename all of the files with a particular file extension in a directory. The script should take 2 arguments: the base name of the files to rename and the file extension. For this problem, use the zipped file Pictures.tar.gz located on the Shared git repository. Note, you should copy the zipped file to your local repository and unzip the file using cp and tar commands



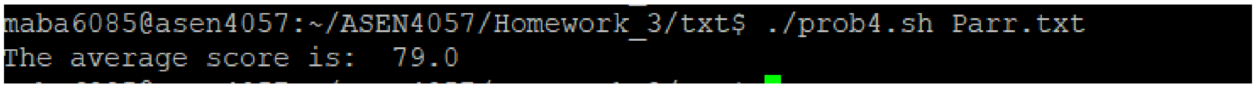
```
maba6085@asen4057:~/ASEN4057/Homework_3/jpg$ ls
Galaxy.jpg  Isles.jpg  Lake.jpg  Lion.jpg  Moon.jpg  prob3.sh  Snow.jpg  Underwater.jpg
maba6085@asen4057:~/ASEN4057/Homework_3/jpg$ ./prob3.sh MyPictures jpg
maba6085@asen4057:~/ASEN4057/Homework_3/jpg$ ls
MyPictures001.jpg  MyPictures002.jpg  MyPictures003.jpg  MyPictures004.jpg  MyPictures005.jpg  MyPictures006.jpg  MyPictures007.jpg  prob3.sh
maba6085@asen4057:~/ASEN4057/Homework_3/jpg$
```

Script:



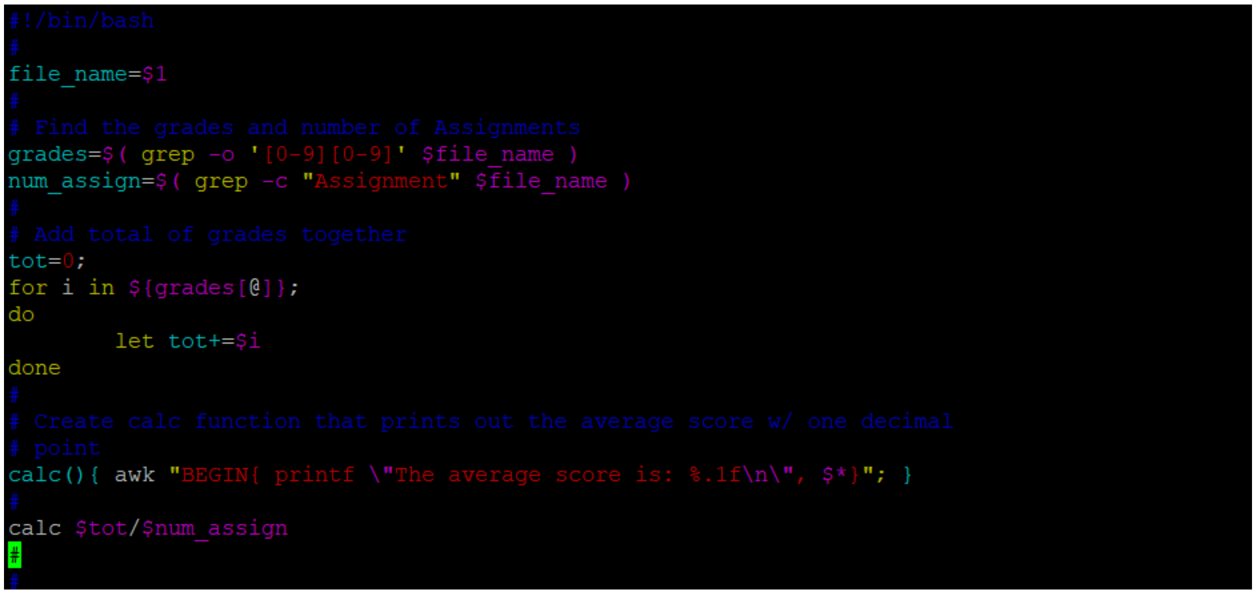
```
#!/bin/bash
#
# This is a shell script for Problem 3 of Homework 3
#
base_name=$1
#
base_file=$2
#
num=1;
#
for i in *.$base_file
do
        mv "$i" "$(printf $base_name'%02d'$num).${i#*.}";
        ((num++))
done
~
```

Problem 4:
Write a Bash shell script called prob4.sh that computes the average grade from a grade text file. The grade file will contain multiple lines and each line with contain a grade and its descriptive information. These items will be separated by a semicolon. You need to read the file line-by-line, parse the grade of each file, and sum them up to get the total score. The grade in each line will be an integer.



```
maba6085@asen4057:~/ASEN4057/Homework_3/txt$ ./prob4.sh Parr.txt
The average score is:  79.0
```

Script:

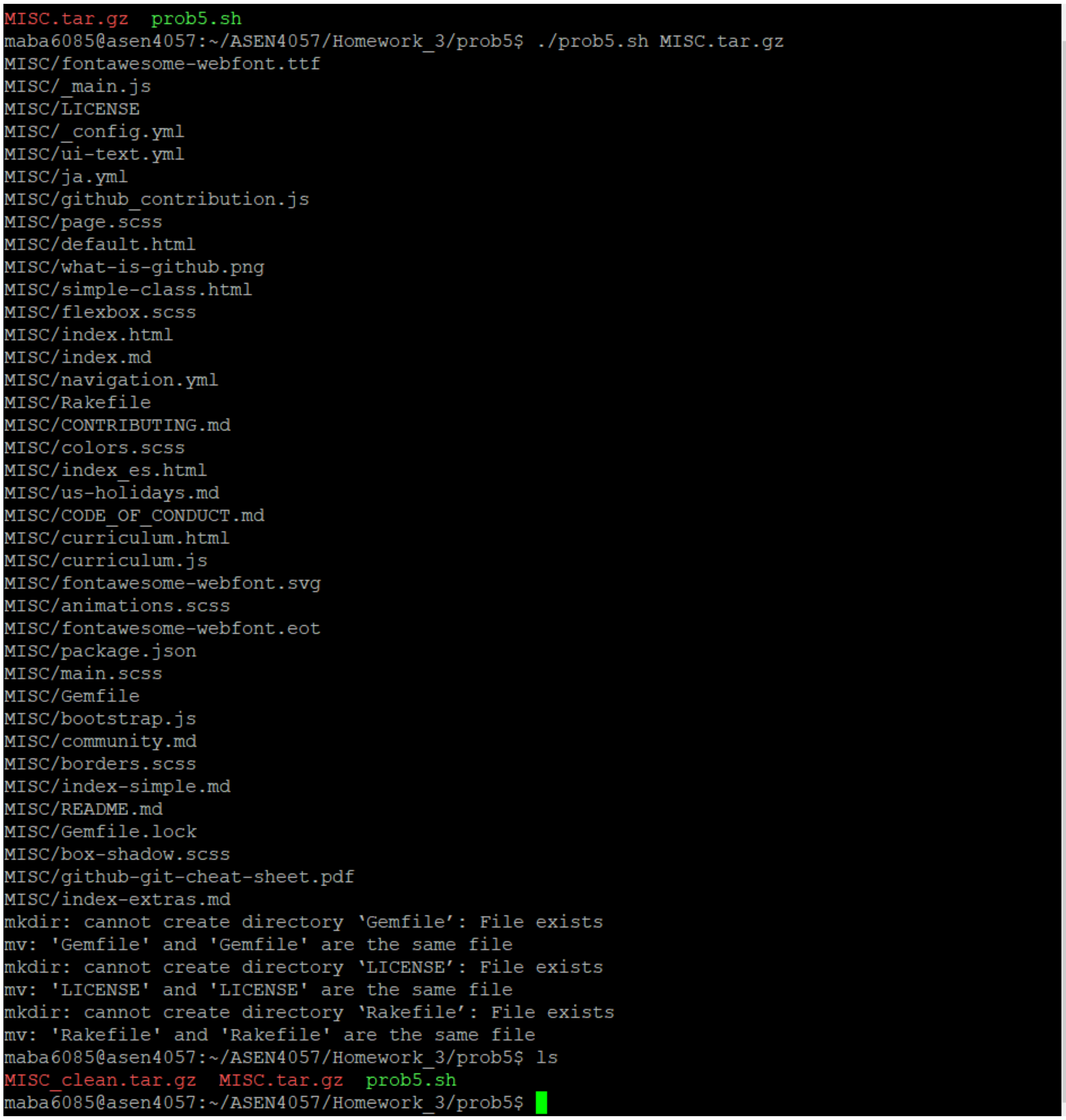


```
#!/bin/bash
#
file_name=$1
#
# Find the grades and number of Assignments
grades=$( grep -o '[0-9][0-9]' $file_name )
num_assign=$( grep -c "Assignment" $file_name )
#
# Add total of grades together
tot=0;
for i in ${grades[@]};
do
        let tot+=$i
done
#
# Create calc function that prints out the average score w/ one decimal
# point
calc(){ awk "BEGIN{ printf \"The average score is: %.1f\n\", $*}"; }
#
calc $tot/$num_assign
#
#
```

Problem 5:

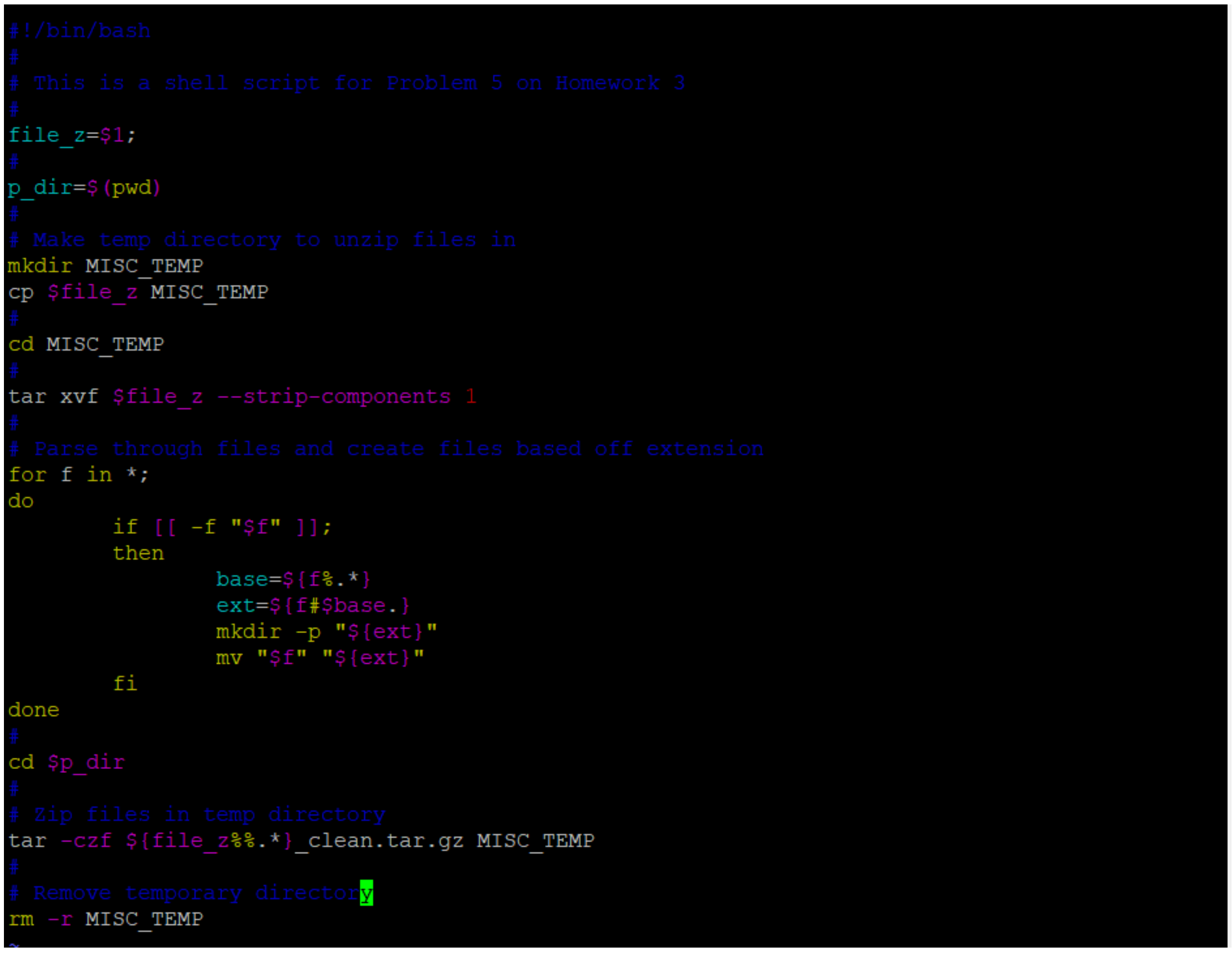
Write a Bash shell script called prob5.sh which organizes the files within a zipped file. Your script should take in as an input, the name of the zipped file to organize. The script should perform the following tasks:

1. Unzip the file
2. Create subfolders for each file extension associated with the extracted files. Place the files within subdirectories associated with their particular file extensions. If a file does not have a file extension, it should not be placed within any particular subdirectory.
3. Zip the resulting folder structure (excluding the original zipped file). This new zipped file should have the identifier [NAME]_clean within its name.



```
MISC.tar.gz  prob5.sh
maba6085@asen4057:~/ASEN4057/Homework_3/prob5$ ./prob5.sh MISC.tar.gz
MISC/fontawesome-webfont.ttf
MISC/_main.js
MISC/LICENSE
MISC/_config.yml
MISC/ui-text.yml
MISC/ja.yml
MISC/github_contribution.js
MISC/page.scss
MISC/default.html
MISC/what-is-github.png
MISC/simple-class.html
MISC/flexbox.scss
MISC/index.html
MISC/index.md
MISC/navigation.yml
MISC/Rakefile
MISC/CONTRIBUTING.md
MISC/colors.scss
MISC/index_es.html
MISC/us-holidays.md
MISC/CODE_OF_CONDUCT.md
MISC/curriculum.html
MISC/curriculum.js
MISC/fontawesome-webfont.svg
MISC/animations.scss
MISC/fontawesome-webfont.eot
MISC/package.json
MISC/main.scss
MISC/Gemfile
MISC/bootstrap.js
MISC/community.md
MISC/borders.scss
MISC/index-simple.md
MISC/README.md
MISC/Gemfile.lock
MISC/box-shadow.scss
MISC/github-git-cheat-sheet.pdf
MISC/index-extras.md
mkdir: cannot create directory ‘Gemfile’: File exists
mv: 'Gemfile' and 'Gemfile' are the same file
mkdir: cannot create directory ‘LICENSE’: File exists
mv: 'LICENSE' and 'LICENSE' are the same file
mkdir: cannot create directory ‘Rakefile’: File exists
mv: 'Rakefile' and 'Rakefile' are the same file
maba6085@asen4057:~/ASEN4057/Homework_3/prob5$ ls
MISC_clean.tar.gz  MISC.tar.gz  prob5.sh
maba6085@asen4057:~/ASEN4057/Homework_3/prob5$
```

Script:



```
#!/bin/bash
#
# This is a shell script for Problem 5 on Homework 3
#
file_z=$1;
#
p_dir=$(pwd)
#
# Make temp directory to unzip files in
mkdir MISC_TEMP
cp $file_z MISC_TEMP
#
cd MISC_TEMP
#
tar xvf $file_z --strip-components 1
#
# Parse through files and create files based off extension
for f in *;
do
        if [[ -f "$f" ]];
        then
                base=${f%.*}
                ext=${f#$base.}
                mkdir -p "${ext}"
                mv "$f" "${ext}"
        fi
done
#
cd $p_dir
#
# Zip files in temp directory
tar -czf ${file_z%%.*}_clean.tar.gz MISC_TEMP
#
# Remove temporary directory
rm -r MISC_TEMP
```