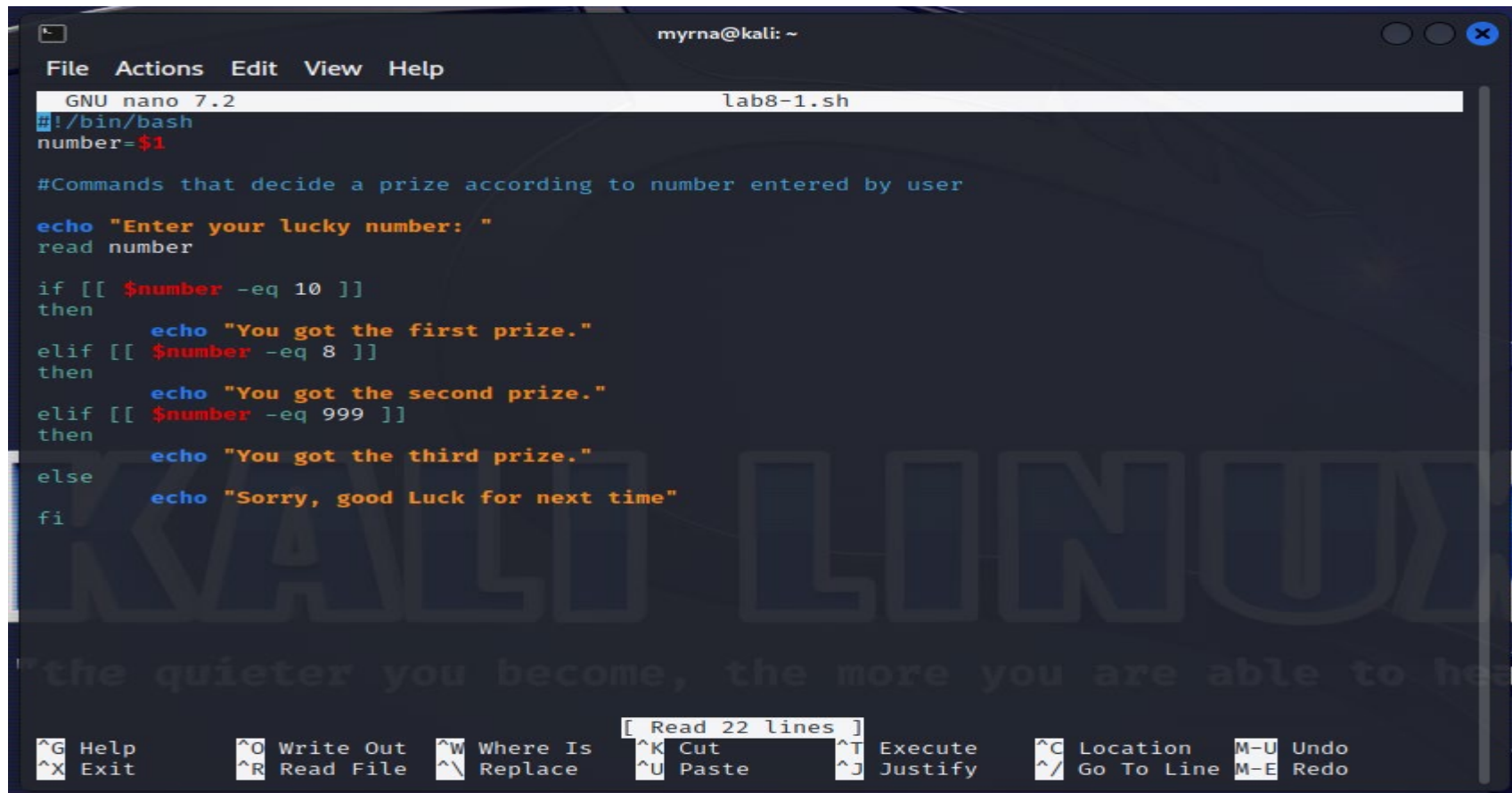


Myrna E. Santiago

CYSE 270_20306

October 31, 2023

Task A, Step 1: A shell script is written using nano with if-elif-else conditions to determine if a person won one of three prizes according to their favorite number.




```
myrna@kali: ~  
File Actions Edit View Help  
GNU nano 7.2 lab8-1.sh  
#!/bin/bash  
number=$1  
  
#Commands that decide a prize according to number entered by user  
  
echo "Enter your lucky number: "  
read number  
  
if [[ $number -eq 10 ]]  
then  
    echo "You got the first prize."  
elif [[ $number -eq 8 ]]  
then  
    echo "You got the second prize."  
elif [[ $number -eq 999 ]]  
then  
    echo "You got the third prize."  
else  
    echo "Sorry, good Luck for next time"  
fi
```

[Read 22 lines]

^G Help	^O Write Out	^W Where Is	^K Cut	^T Execute	^C Location	M-U Undo
^X Exit	^R Read File	^\\ Replace	^U Paste	^J Justify	^/_ Go To Line	M-E Redo

Step 2: Using `ls -l lab8-1.sh` we verify the permissions of the file. Using `chmod a+x lab8-1.sh` we add executing privileges to users and verify again that it is done with `ls -l`.



```
myrna@kali: ~  
File Actions Edit View Help  
  
(myrna@kali)-[~]  
$ nano lab8-1.sh  
  
(myrna@kali)-[~]  
$ ls -l lab8-1.sh  
-rw-r--r-- 1 myrna myrna 372 Oct 31 19:12 lab8-1.sh  
  
(myrna@kali)-[~]  
$ chmod a+x lab8-1.sh  
  
(myrna@kali)-[~]  
$ ls -l lab8-1.sh  
-rwxr-xr-x 1 myrna myrna 372 Oct 31 19:12 lab8-1.sh  
  
(myrna@kali)-[~]  
$
```

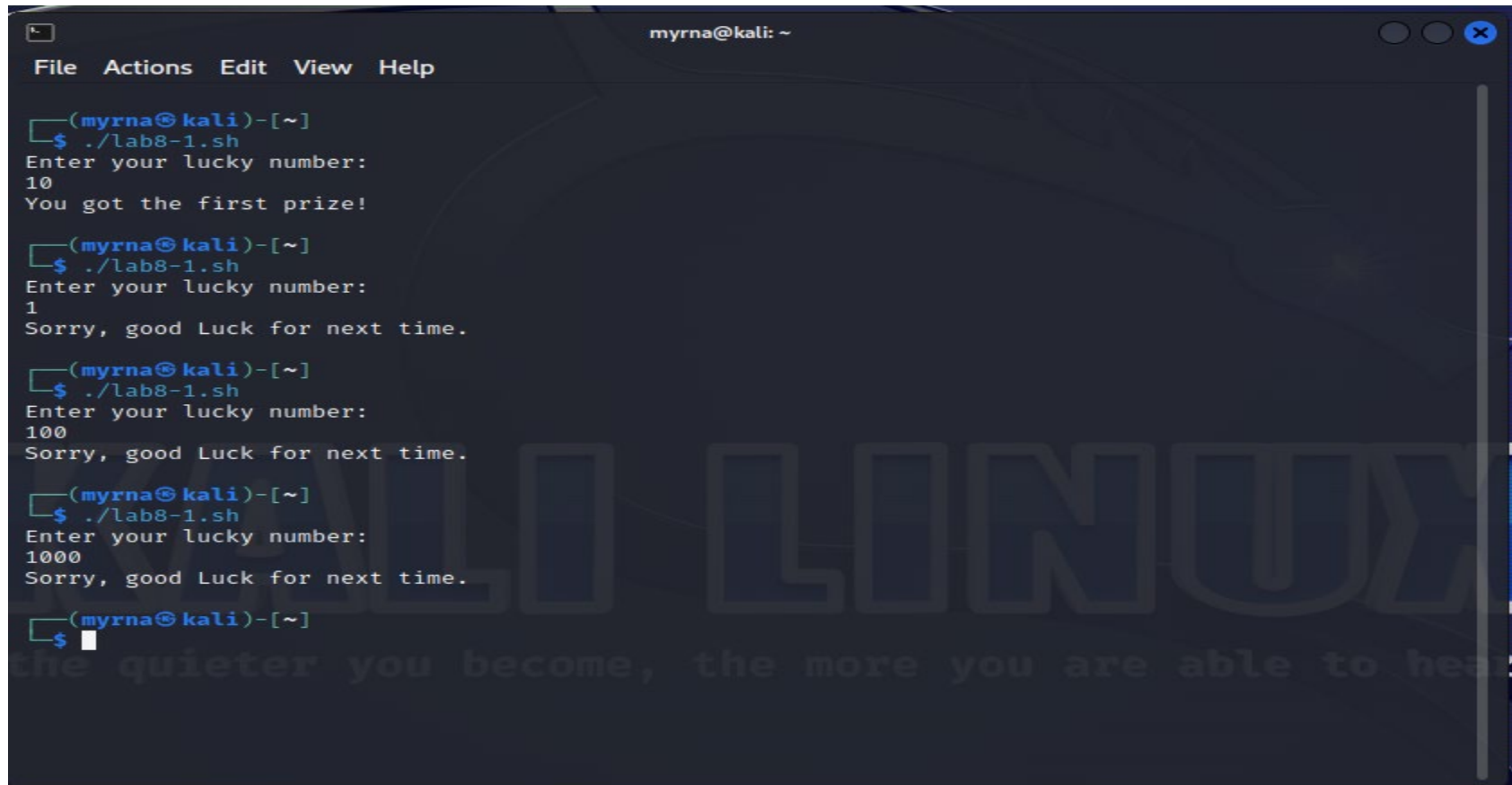
KALI LINUX
the quieter you become, the more you are able to hear

Step 3: We use cat lab8-1.sh to read the file content and make sure the content is correct.

A terminal window titled 'myrna@kali: ~' with a menu bar (File, Actions, Edit, View, Help). The terminal shows the command 'cat lab8-1.sh' being executed. The output is a shell script that prompts a user for a lucky number and checks it against three prize conditions: 10 (first prize), 8 (second prize), and 999 (third prize). If none of these conditions are met, it says 'Sorry, good Luck for next time.' The prompt '(myrna@kali)-[~]' is shown at the top and bottom of the terminal output.

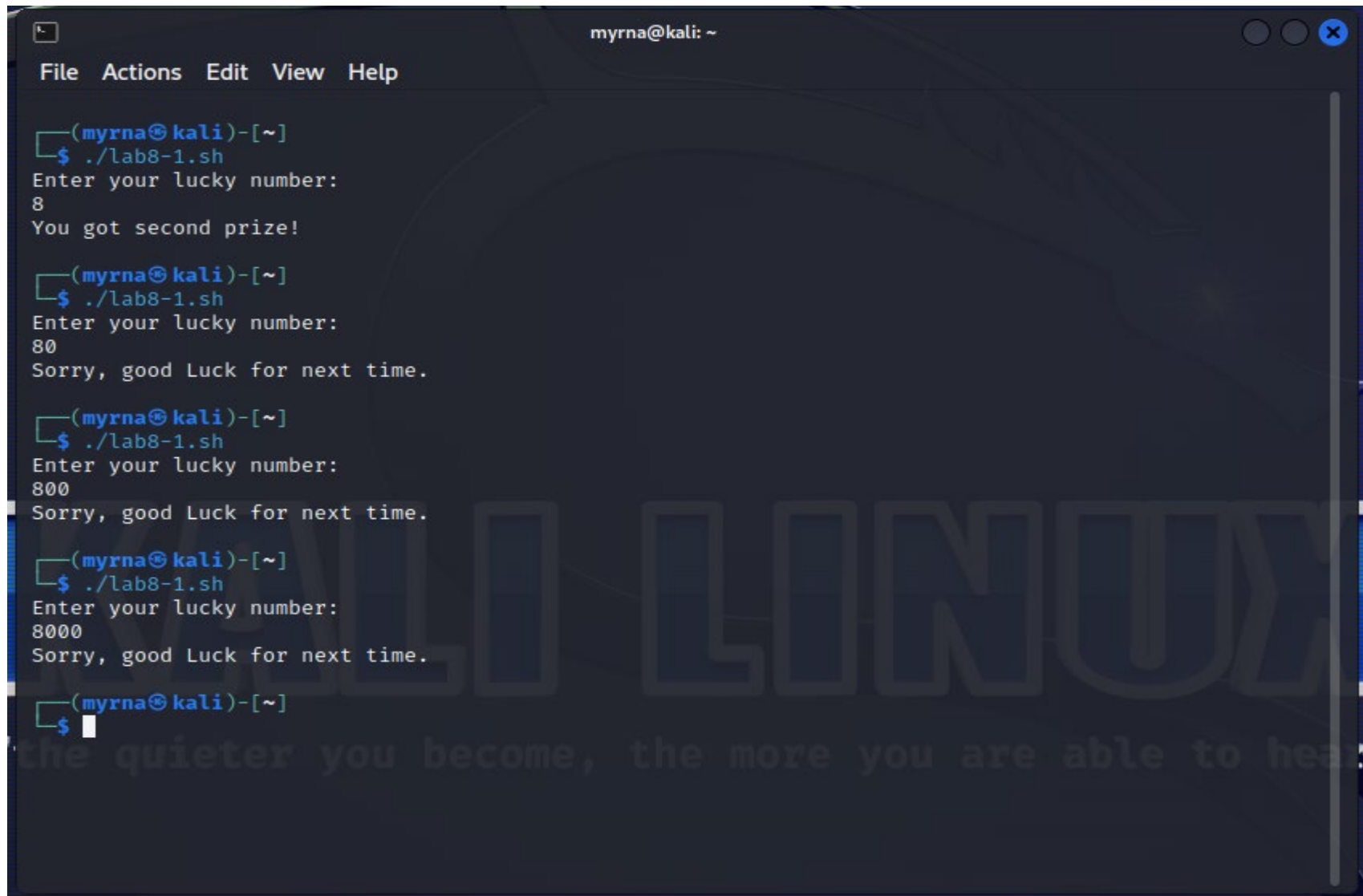
```
(myrna@kali)-[~]  
$ cat lab8-1.sh  
#!/bin/bash  
number=$1  
  
#User is prompted to enter their favorite number.  
  
echo "Enter your lucky number:"  
read number  
  
#Commands compare the number entered and decide if the number match one of the prizes.  
  
if [[ $number -eq 10 ]]  
then  
    echo "You got the first prize!"  
elif [[ $number -eq 8 ]]  
then  
    echo "You got second prize!"  
elif [[ $number -eq 999 ]]  
then  
    echo "You got the third prize!"  
else  
    echo "Sorry, good Luck for next time."  
fi  
  
(myrna@kali)-[~]  
$
```

Step 4 a-b-c: A test is performed using the correct numbers, and also similar ones to make sure that the script recognizes the correct ones and reject the others.



```
myrna@kali: ~  
File Actions Edit View Help  
(myrna@kali)-[~]  
$ ./lab8-1.sh  
Enter your lucky number:  
10  
You got the first prize!  
(myrna@kali)-[~]  
$ ./lab8-1.sh  
Enter your lucky number:  
1  
Sorry, good Luck for next time.  
(myrna@kali)-[~]  
$ ./lab8-1.sh  
Enter your lucky number:  
100  
Sorry, good Luck for next time.  
(myrna@kali)-[~]  
$ ./lab8-1.sh  
Enter your lucky number:  
1000  
Sorry, good Luck for next time.  
(myrna@kali)-[~]  
$
```

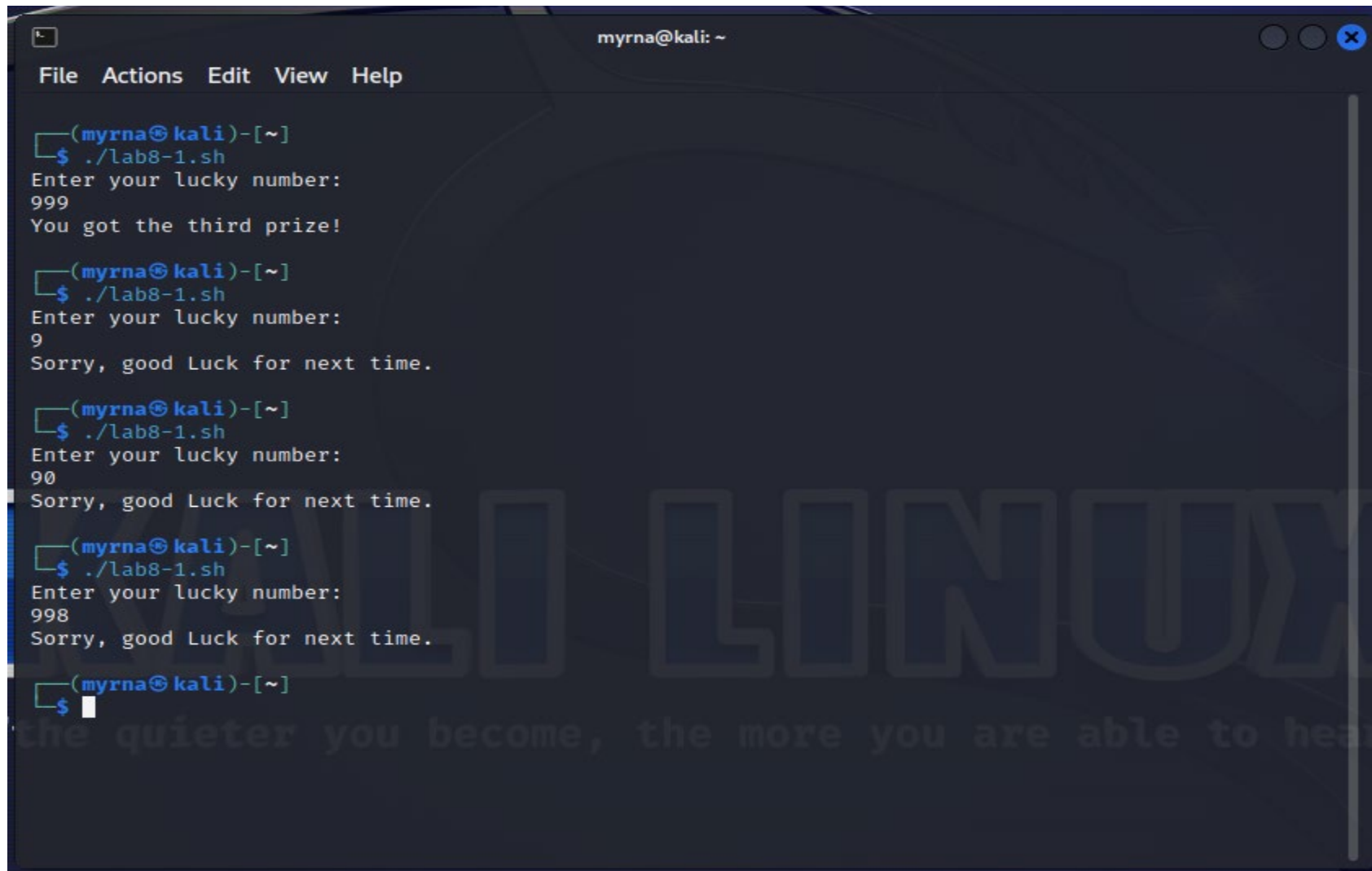
Step 4b:



A terminal window titled "myrna@kali: ~" with a menu bar containing "File", "Actions", "Edit", "View", and "Help". The terminal shows four consecutive runs of the script `./lab8-1.sh`. Each run prompts the user to "Enter your lucky number:". The first run uses the number 8 and outputs "You got second prize!". The second run uses 80 and outputs "Sorry, good Luck for next time.". The third run uses 800 and outputs "Sorry, good Luck for next time.". The fourth run uses 8000 and outputs "Sorry, good Luck for next time.". The prompt is currently at the start of a new line.

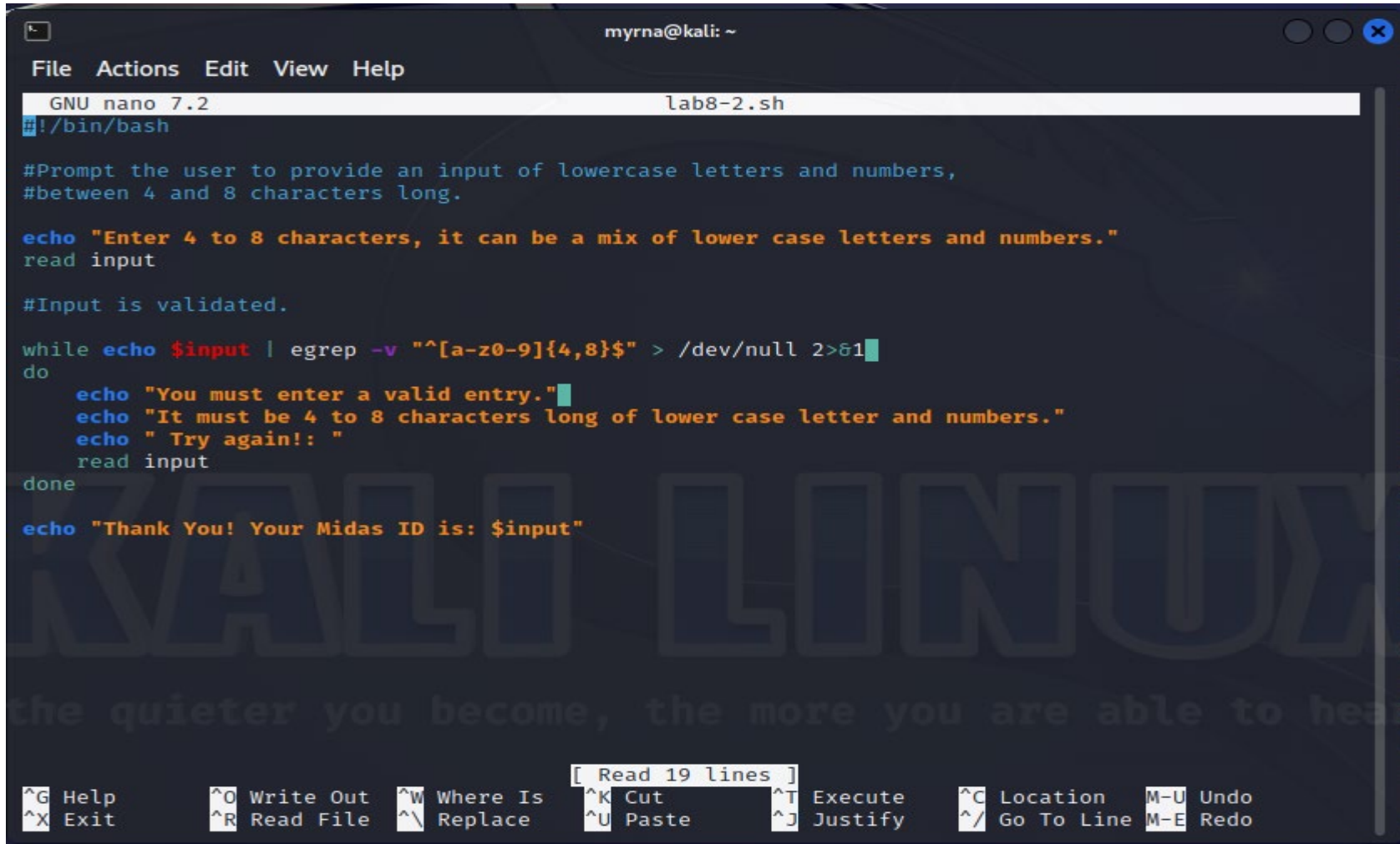
```
(myrna@kali)-[~]  
$ ./lab8-1.sh  
Enter your lucky number:  
8  
You got second prize!  
  
(myrna@kali)-[~]  
$ ./lab8-1.sh  
Enter your lucky number:  
80  
Sorry, good Luck for next time.  
  
(myrna@kali)-[~]  
$ ./lab8-1.sh  
Enter your lucky number:  
800  
Sorry, good Luck for next time.  
  
(myrna@kali)-[~]  
$ ./lab8-1.sh  
Enter your lucky number:  
8000  
Sorry, good Luck for next time.  
  
(myrna@kali)-[~]  
$
```

Step 4c:



```
myrna@kali: ~  
File Actions Edit View Help  
  
(myrna@kali)-[~]  
$ ./lab8-1.sh  
Enter your lucky number:  
999  
You got the third prize!  
  
(myrna@kali)-[~]  
$ ./lab8-1.sh  
Enter your lucky number:  
9  
Sorry, good Luck for next time.  
  
(myrna@kali)-[~]  
$ ./lab8-1.sh  
Enter your lucky number:  
90  
Sorry, good Luck for next time.  
  
(myrna@kali)-[~]  
$ ./lab8-1.sh  
Enter your lucky number:  
998  
Sorry, good Luck for next time.  
  
(myrna@kali)-[~]  
$
```

Task B Step 1: Using nano editor we wrote a script to ask a user for an input of 4 to 8 characters of lower-case letters and numbers 0 to 9. A while loop is use with the command egrep to perform the task.



```
myrna@kali: ~
File Actions Edit View Help
GNU nano 7.2 lab8-2.sh
#!/bin/bash

#Prompt the user to provide an input of lowercase letters and numbers,
#between 4 and 8 characters long.

echo "Enter 4 to 8 characters, it can be a mix of lower case letters and numbers."
read input

#Input is validated.

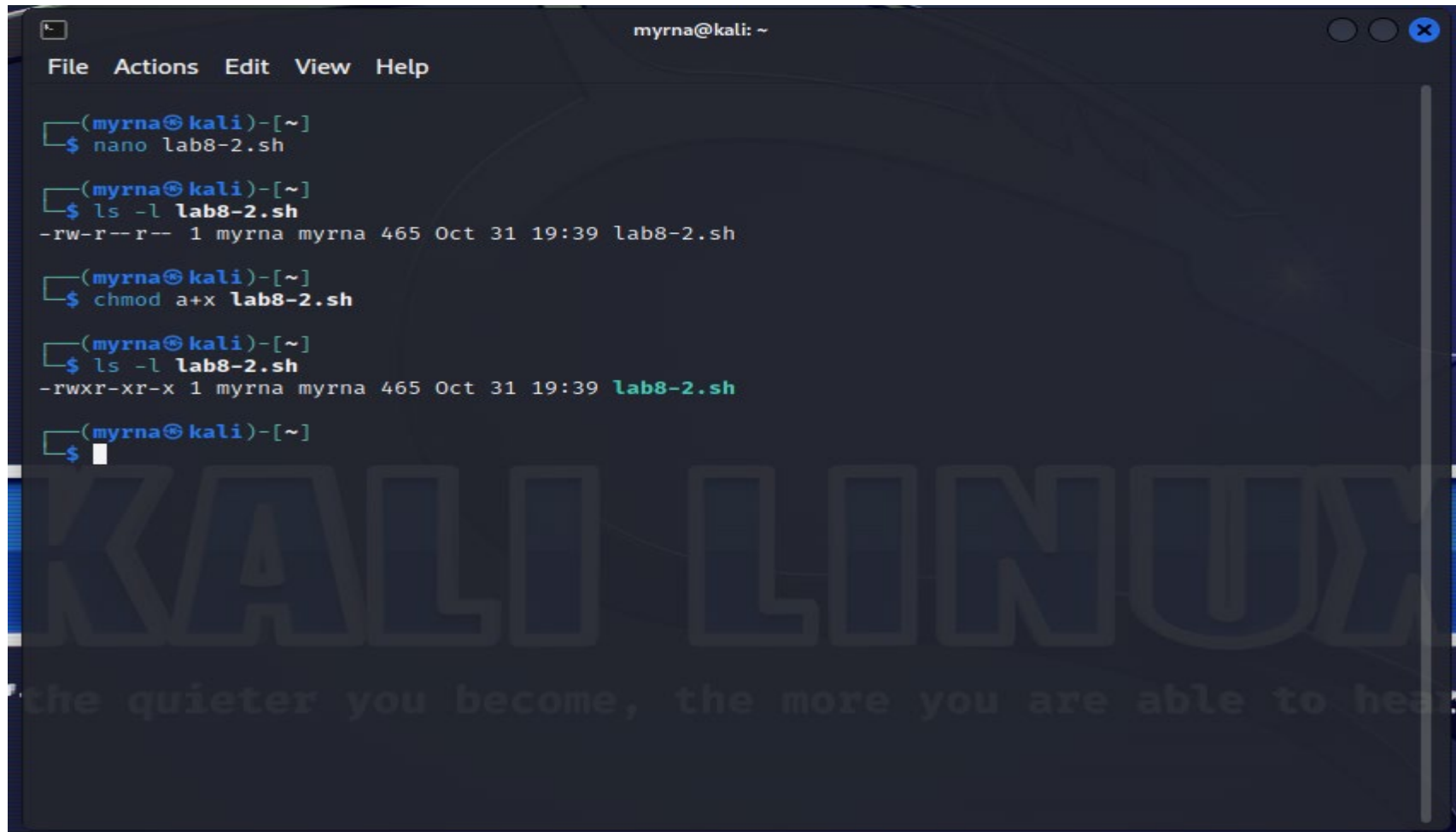
while echo $input | egrep -v "^[a-z0-9]{4,8}$" > /dev/null 2>&1
do
    echo "You must enter a valid entry."
    echo "It must be 4 to 8 characters long of lower case letter and numbers."
    echo " Try again!: "
    read input
done

echo "Thank You! Your Midas ID is: $input"
```

[Read 19 lines]

^G Help	^O Write Out	^W Where Is	^K Cut	^T Execute	^C Location	M-U Undo
^X Exit	^R Read File	^_ Replace	^U Paste	^J Justify	^_ Go To Line	M-E Redo

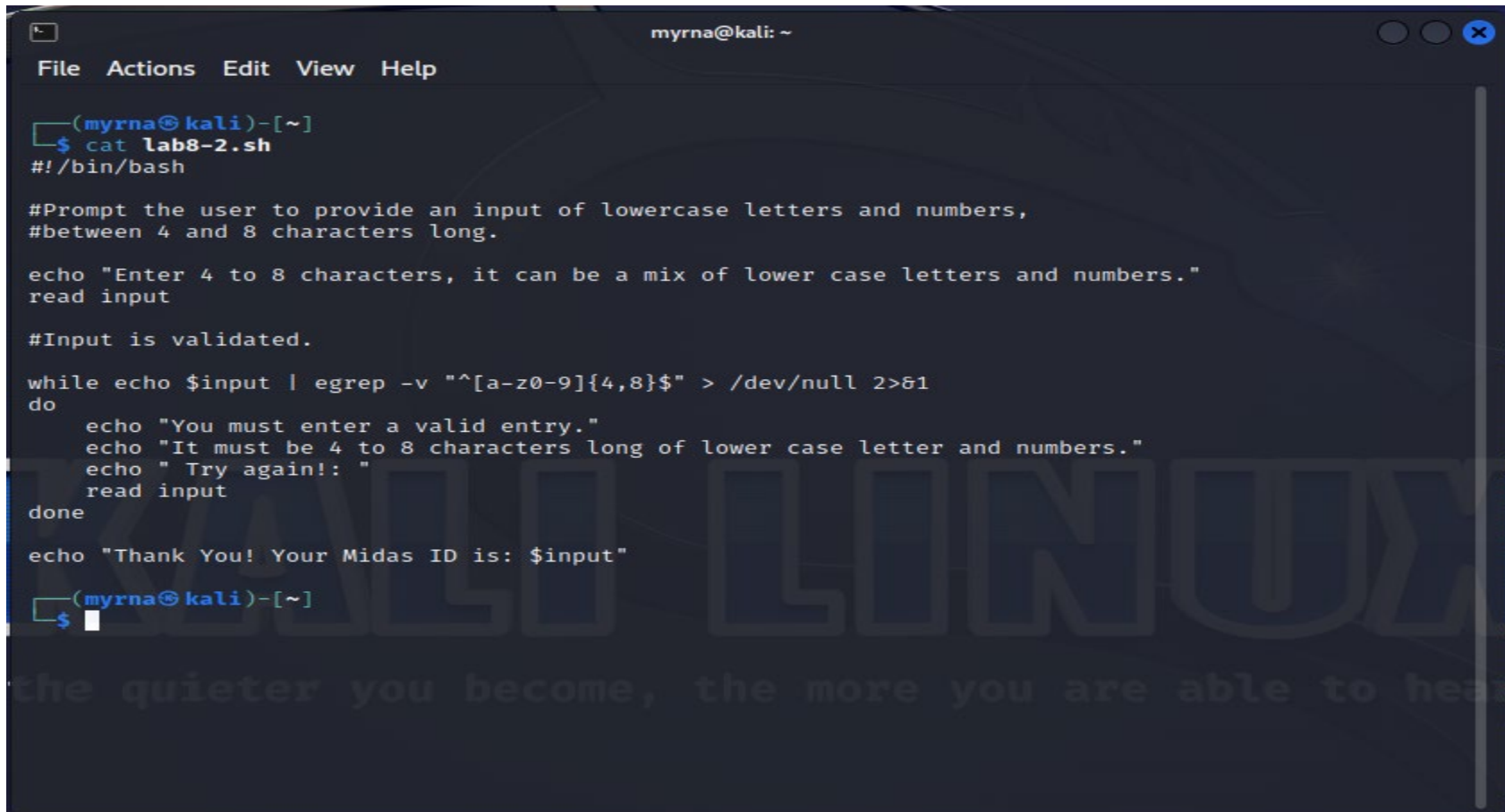
Step 2: Using `ls -l lab8-2.sh` we verify the user rights to the file and we modified it using `chmod a+x lab8-2.sh`. Then we verify one more time that the changes took effect.



```
myrna@kali: ~  
File Actions Edit View Help  
(myrna@kali)-[~]  
$ nano lab8-2.sh  
(myrna@kali)-[~]  
$ ls -l lab8-2.sh  
-rw-r--r-- 1 myrna myrna 465 Oct 31 19:39 lab8-2.sh  
(myrna@kali)-[~]  
$ chmod a+x lab8-2.sh  
(myrna@kali)-[~]  
$ ls -l lab8-2.sh  
-rwxr-xr-x 1 myrna myrna 465 Oct 31 19:39 lab8-2.sh  
(myrna@kali)-[~]  
$
```

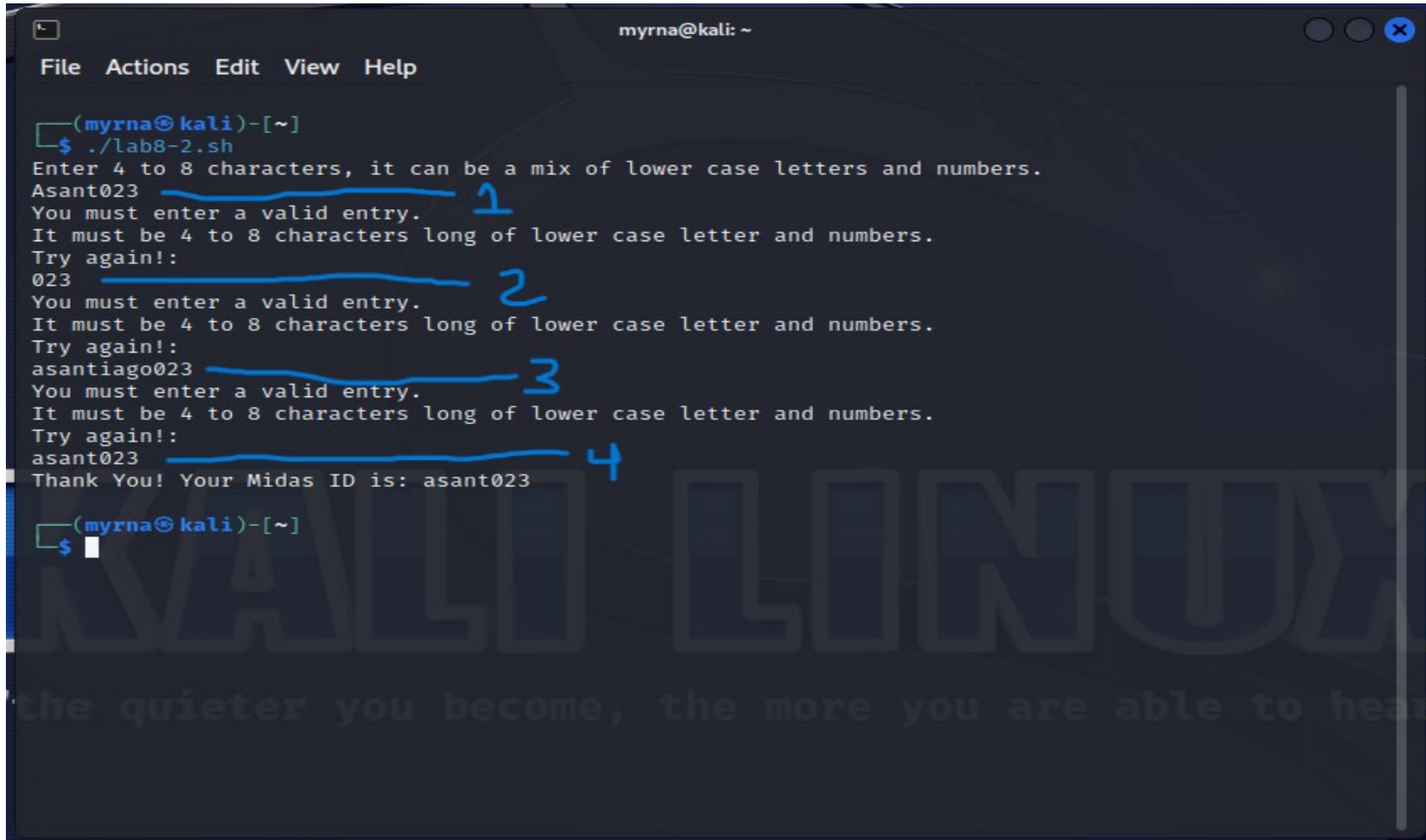
A terminal window titled "myrna@kali: ~" with a menu bar (File, Actions, Edit, View, Help). The terminal shows a series of commands and their outputs. First, the user runs `nano lab8-2.sh`. Then, they run `ls -l lab8-2.sh`, which shows the file permissions as `-rw-r--r--`. Next, they run `chmod a+x lab8-2.sh`. Finally, they run `ls -l lab8-2.sh` again, and the permissions are now `-rwxr-xr-x`, with the filename `lab8-2.sh` highlighted in green. A large, semi-transparent "KALI LINUX" watermark is visible across the bottom half of the terminal window.

Step 3: Using cat lab8-2.sh, we verify the content of the file.

A terminal window titled 'myrna@kali: ~' with a menu bar (File, Actions, Edit, View, Help). The terminal shows the command 'cat lab8-2.sh' being executed. The output is a shell script that prompts the user for a 4-8 character alphanumeric input, validates it using a while loop and egrep, and prints the ID if valid. A large 'KALI LINUX' watermark is visible across the terminal content.

```
(myrna@kali)-[~]  
$ cat lab8-2.sh  
#!/bin/bash  
  
#Prompt the user to provide an input of lowercase letters and numbers,  
#between 4 and 8 characters long.  
  
echo "Enter 4 to 8 characters, it can be a mix of lower case letters and numbers."  
read input  
  
#Input is validated.  
  
while echo $input | egrep -v "^[a-z0-9]{4,8}$" > /dev/null 2>&1  
do  
    echo "You must enter a valid entry."  
    echo "It must be 4 to 8 characters long of lower case letter and numbers."  
    echo " Try again!: "  
    read input  
done  
  
echo "Thank You! Your Midas ID is: $input"  
  
(myrna@kali)-[~]  
$
```

Step 4: We test the program 4 times. First test we added an uppercase letter and the program correctly ask for a valid entry. The second test we added less than 4 digits, again the program asked for a valid entry. The third test we added more than 8 characters and the program one more time asked for a valid entry. Finally, we enter the correct number of characters with corresponding types and the program gave us a correct output.



```
myrna@kali: ~  
File Actions Edit View Help  
  
(myrna@kali)-[~]  
$ ./lab8-2.sh  
Enter 4 to 8 characters, it can be a mix of lower case letters and numbers.  
Asant023 1  
You must enter a valid entry.  
It must be 4 to 8 characters long of lower case letter and numbers.  
Try again!:  
023 2  
You must enter a valid entry.  
It must be 4 to 8 characters long of lower case letter and numbers.  
Try again!:  
asantiago023 3  
You must enter a valid entry.  
It must be 4 to 8 characters long of lower case letter and numbers.  
Try again!:  
asant023 4  
Thank You! Your Midas ID is: asant023  
  
(myrna@kali)-[~]  
$
```