Name: Anand Kumar Gupta   Email id: ananddkg99.9@gmail.com

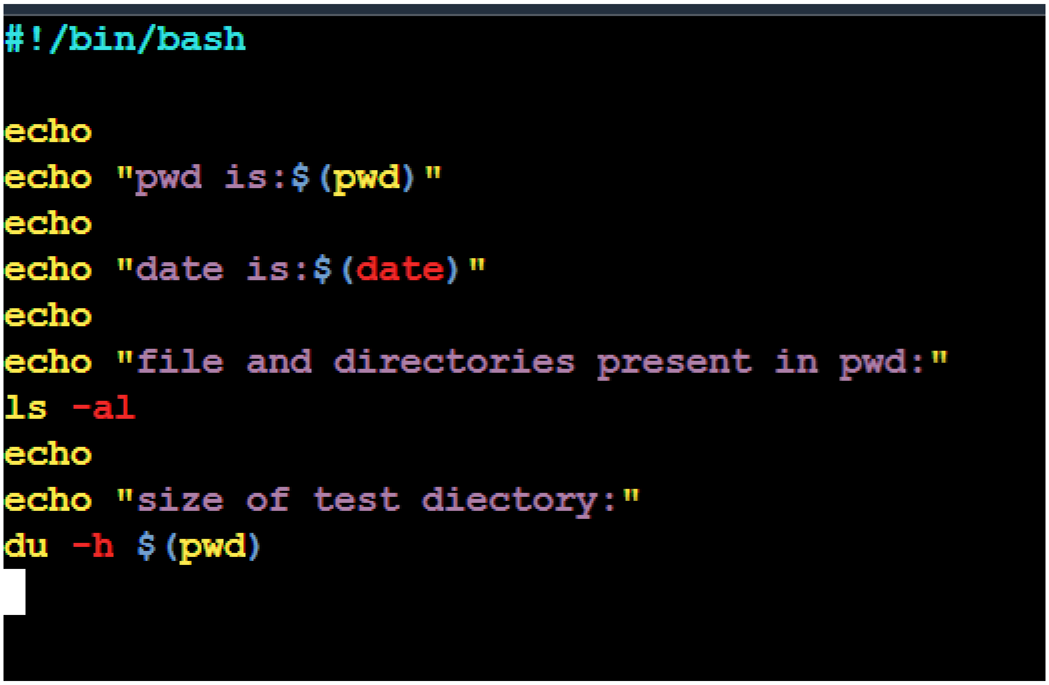
Assignment 5

Q1) How to upload HTML web pages on Apache2 web server in EC-2 Instance? Please justify with step by step answers.

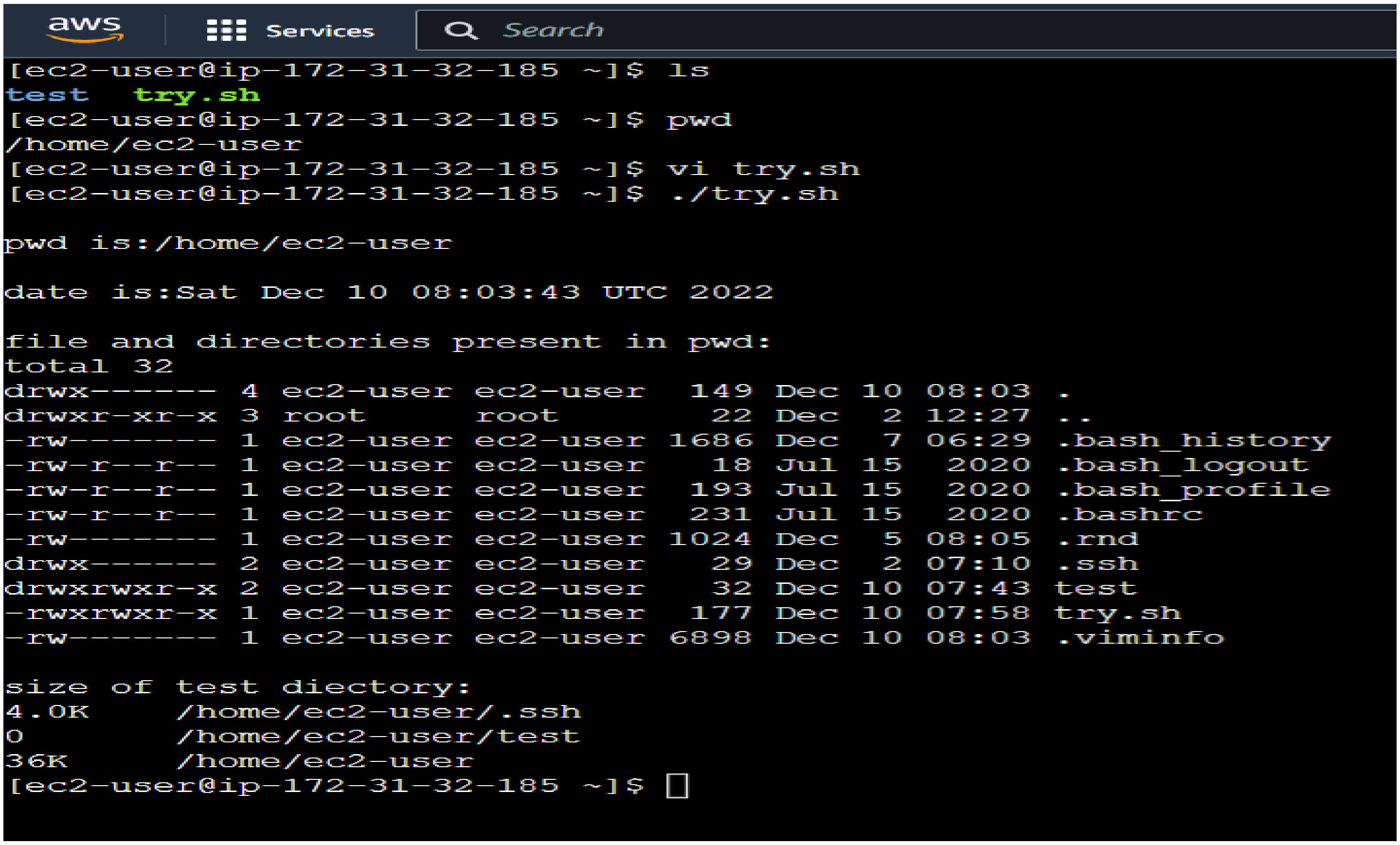
Soln: First launch a AWS Linux based instance and then setup inbound and outbound for port 80. Then type following commands in terminal:-

1) sudo su –
2) yum upate –y
3) yum install httpd
4) systemctl start httpd
5) httpd (to check status)
6) systemctl enable httpd (so that we need not start httpd server again and again when we connect to instance)
7) echo “any text” > /var/www/html/index.html

Q2) Create readfile.sh in which you can read the information of PWD like size, permission, date time etc.

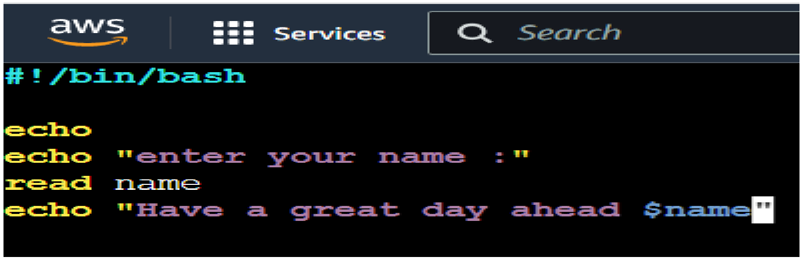


```
#!/bin/bash

echo
echo "pwd is:$(pwd)"
echo
echo "date is:$(date)"
echo
echo "file and directories present in pwd:"
ls -al
echo
echo "size of test diectory:"
du -h $(pwd)
```

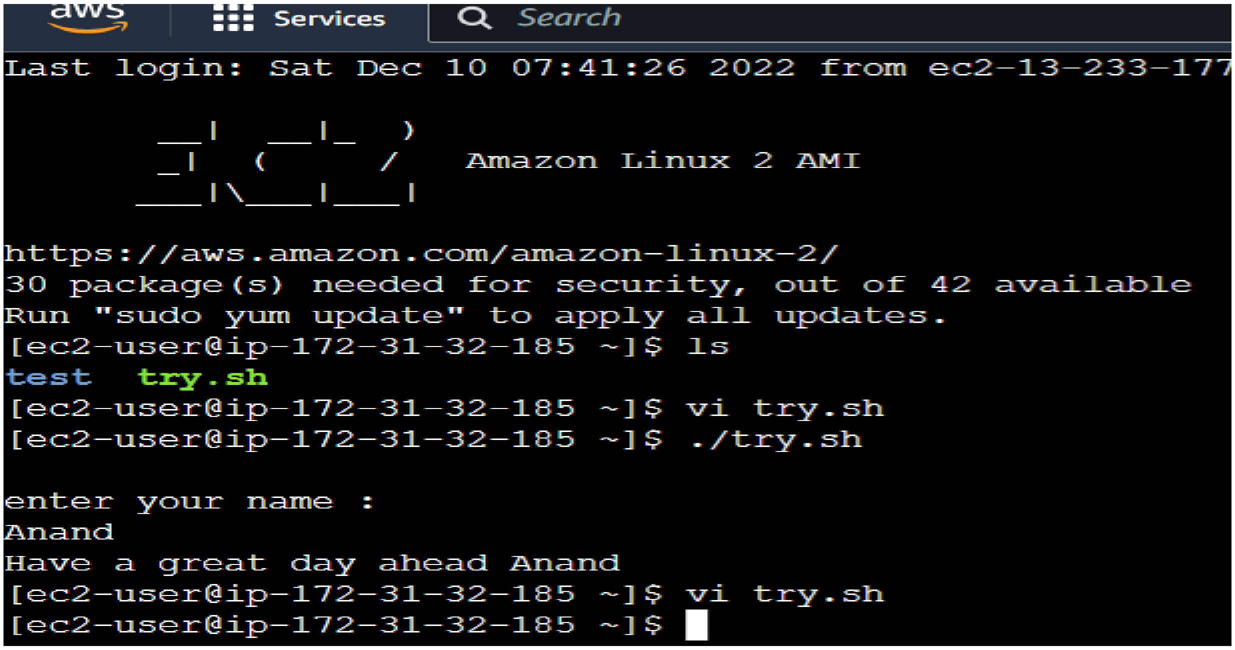


```
[ec2-user@ip-172-31-32-185 ~]$ ls
test  try.sh
[ec2-user@ip-172-31-32-185 ~]$ pwd
/home/ec2-user
[ec2-user@ip-172-31-32-185 ~]$ vi try.sh
[ec2-user@ip-172-31-32-185 ~]$ ./try.sh

pwd is:/home/ec2-user

date is:Sat Dec 10 08:03:43 UTC 2022

file and directories present in pwd:
total 32
drwx------ 4 ec2-user ec2-user  149 Dec 10 08:03 .
drwxr-xr-x 3 root     root       22 Dec  2 12:27 ..
-rw------- 1 ec2-user ec2-user 1686 Dec  7 06:29 .bash_history
-rw-r--r-- 1 ec2-user ec2-user   18 Jul 15  2020 .bash_logout
-rw-r--r-- 1 ec2-user ec2-user  193 Jul 15  2020 .bash_profile
-rw-r--r-- 1 ec2-user ec2-user  231 Jul 15  2020 .bashrc
-rw------- 1 ec2-user ec2-user 1024 Dec  5 08:05 .rnd
drwx------ 2 ec2-user ec2-user   29 Dec  2 07:10 .ssh
drwxrwxr-x 2 ec2-user ec2-user   32 Dec 10 07:43 test
-rwxrwxr-x 1 ec2-user ec2-user  177 Dec 10 07:58 try.sh
-rw------- 1 ec2-user ec2-user 6898 Dec 10 08:03 .viminfo

size of test diectory:
4.0K    /home/ec2-user/.ssh
0       /home/ec2-user/test
36K     /home/ec2-user
[ec2-user@ip-172-31-32-185 ~]$ 
```

Q3) Take an input of name from user and print Have a great day ahead {name}



```
#!/bin/bash

echo
echo "enter your name :"
read name
echo "Have a great day ahead $name"
```



```
Last login: Sat Dec 10 07:41:26 2022 from ec2-13-233-177

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

https://aws.amazon.com/amazon-linux-2/
30 package(s) needed for security, out of 42 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-32-185 ~]$ ls
test  try.sh
[ec2-user@ip-172-31-32-185 ~]$ vi try.sh
[ec2-user@ip-172-31-32-185 ~]$ ./try.sh

enter your name :
Anand
Have a great day ahead Anand
[ec2-user@ip-172-31-32-185 ~]$ vi try.sh
[ec2-user@ip-172-31-32-185 ~]$ 
```

Q4) Let’s take a scenario of fintech app program in which we want to have three separate outputs for 3 different situations:

-The balance is less than zero

-The balance is zero

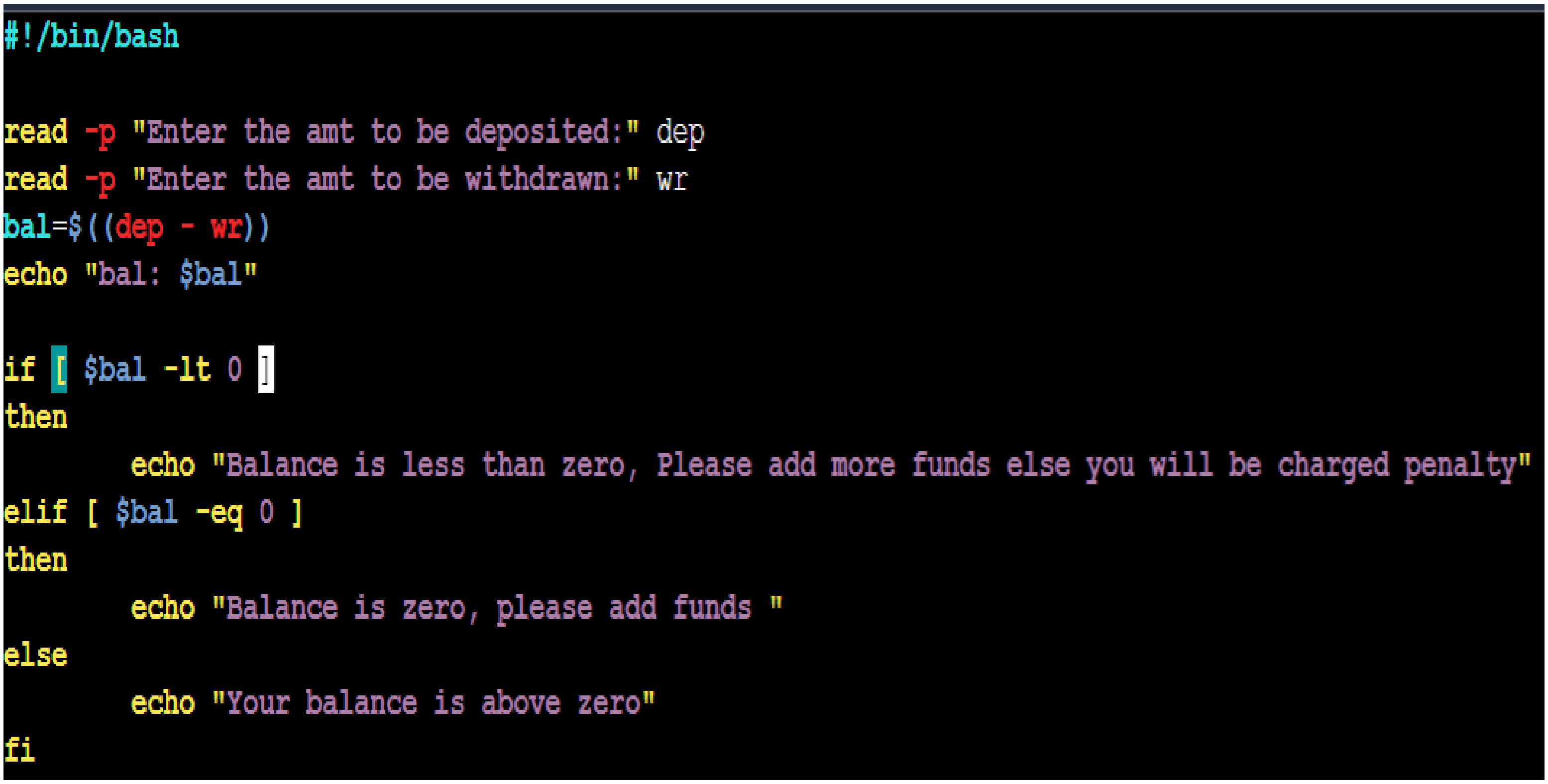
- The balance is above zero

For instance, in the following program, use then if, elif, else statements to display different outputs in different scenarios:
Use “if” condition to check if the balance is less than zero.

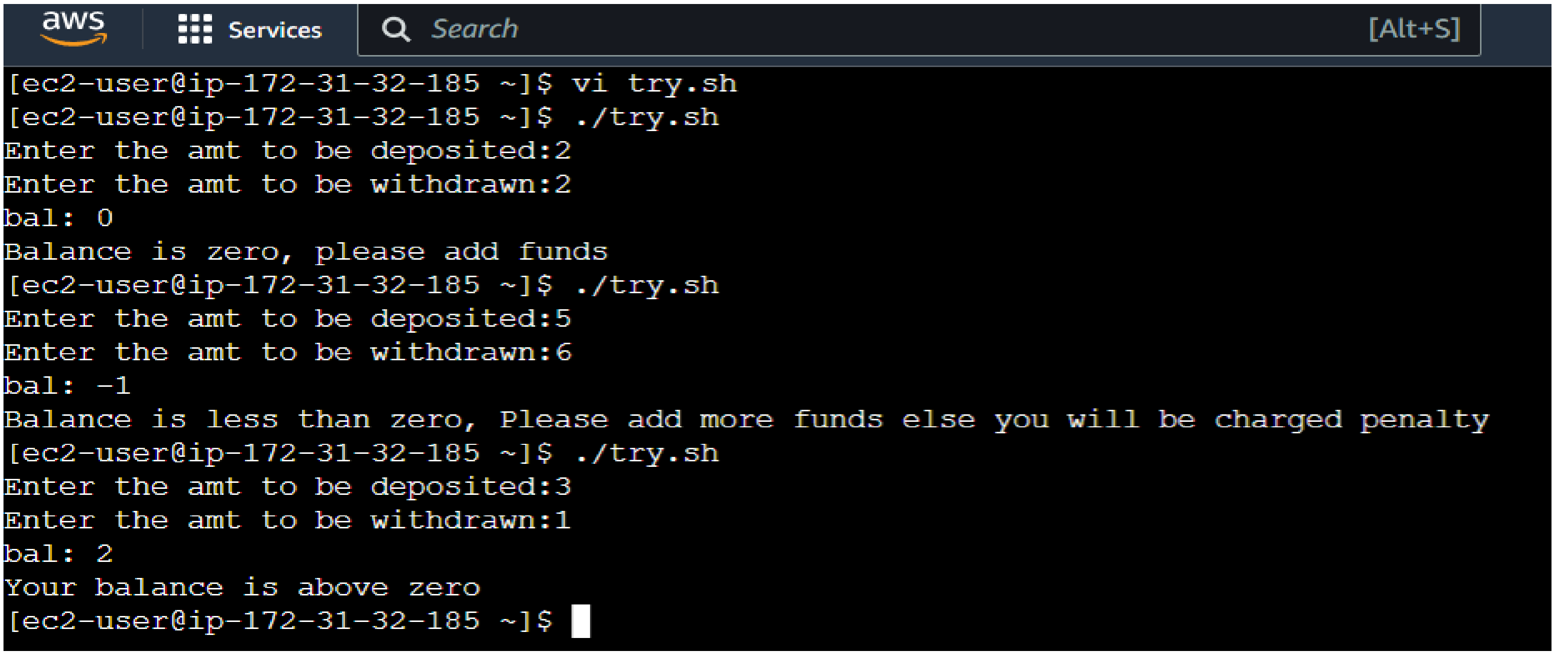
If this condition evaluates to true, display the message using the echo command: “Balance is less than zero, Please add more funds else you will be charged penalty”.

If the above condition does not match, then use “elif” condition to check if the balance is equal to zero. If it evaluates to true, display the message: Balance is zero, please add funds

If none of the above condition matches, use the “else” condition to display the: Your balance is above zero.



```
#!/bin/bash

read -p "Enter the amt to be deposited:" dep
read -p "Enter the amt to be withdrawn:" wr
bal=$((dep - wr))
echo "bal: $bal"

if [ $bal -lt 0 ]
then
        echo "Balance is less than zero, Please add more funds else you will be charged penalty"
elif [ $bal -eq 0 ]
then
        echo "Balance is zero, please add funds "
else
        echo "Your balance is above zero"
fi
```



```
[ec2-user@ip-172-31-32-185 ~]$ vi try.sh
[ec2-user@ip-172-31-32-185 ~]$ ./try.sh
Enter the amt to be deposited:2
Enter the amt to be withdrawn:2
bal: 0
Balance is zero, please add funds
[ec2-user@ip-172-31-32-185 ~]$ ./try.sh
Enter the amt to be deposited:5
Enter the amt to be withdrawn:6
bal: -1
Balance is less than zero, Please add more funds else you will be charged penalty
[ec2-user@ip-172-31-32-185 ~]$ ./try.sh
Enter the amt to be deposited:3
Enter the amt to be withdrawn:1
bal: 2
Your balance is above zero
[ec2-user@ip-172-31-32-185 ~]$
```

Q5) Debug and define briefly about the following program :-

```
#!/bin/bash

# Print a message about disk useage.

space_free=$( df -h | awk '{ print $5 }' | sort -n | tail -n 1 | sed 's/%//' )

case $space_free in

[1-5]*)
echo Plenty of disk space available

[6-7]*)
echo There could be a problem in the near future

8*)
echo Maybe we should look at clearing out old files

9*)
echo We could have a serious problem on our hands soon

*)
echo Something is not quite right here
;;
esac
```

Correct code:

```
#!/bin/bash
space_free=$( df -h | awk '{ print $5 }' | sort -n | tail -n 1 | sed 's/%//' )
#echo "$space_free"
case "$space_free" in
  [1-5]*)
    echo Plenty of disk space available
  ;;
  [6-7]*)
    echo There could be a problem in the near future
  ;;
  8*)
    echo Maybe we should look at clearing out old files
  ;;
  9*)
    echo We could have a serious problem on our hands soon
  ;;
  *)
    echo Something is not quite right here
  ;;
esac
```

Explanation:

This script is checking the available disk space on the system and printing a message depending on the amount of free space. The resulting value is stored in the **space_free** variable.

Here:

df -h command to get information about the disk usage on the system, including the amount of space used and available in human readable format.

awk command to extract the fifth column of the output (the percentage of space used)

sort -n to sort the values numerically (in ascending order)

tail -n 1 to get the last value in the list (the one with the highest percentage of space used)

sed 's/%//' to remove the percent sign

The **case** statement then checks the value of **space_free** and prints a message depending on the range it falls into. For example, if **space_free** is between 1 and 5 (inclusive) i.e (10-50), it will print "Plenty of disk space available". If **space_free** is 9* i.e (90-99), it will print "We could have a serious problem on our hands soon". If the value of **space_free** is not in any of the specified ranges, it will print "Something is not quite right here"