Assignment Sheet 

Question 1: Explain this following bash script:

**#!/bin/bash**

**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**

**Line 1 : shbang is declared to let the kernel know it is a shell command and it has to be interpreted**

**LIne 2 : a variabel is created with name (space\_free name).**

**Df :- report file system disk space usage**

**-h :- print sizes in human readable format (e.g., 1K 234M 2G)**

**(|) :- Pipelines**

**A pipeline is a sequence of one or more commands separated by one of the control operators | or |&. The**

**format for a pipeline is:**

**For ex…**

**[time [-p]] [ ! ] command [ [|⎪|&] command2 ... ]**

**The standard output of command is connected via a pipe to the standard input of command2. This connection is performed before any redirections specified by the command (see REDIRECTION below). If |& is used, the standard error of command is connected to command2's standard input through the pipe; it is shorthand for 2>&1 |. This implicit redirection of the standard error is performed after any redirections specified by the command.**

**The return status of a pipeline is the exit status of the last command, unless the pipefail option is enabled. If pipefail is enabled, the pipeline's return status is the value of the last (rightmost) command to exit with a non-zero status, or zero if all commands exit successfully. If the reserved word ! precedes a pipeline, the exit status of that pipeline is the logical negation of the exit status as described above. The shell waits for all commands in the pipeline to terminate before returning a value. If the time reserved word precedes a pipeline, the elapsed as well as user and system time consumed by**

**its execution are reported when the pipeline terminates. The -p option changes the output format to that specified by POSIX. When the shell is in posix mode, it does not recognize time as a reserved word if the next token begins with a `-'. The TIMEFORMAT variable may be set to a format string that specifies how the timing information should be displayed; see the description of TIMEFORMAT under Shell Variables below.**

**When the shell is in posix mode, time may be followed by a newline. In this case, the shell displays the total user and system time consumed by the shell and its children. The TIMEFORMAT variable may be used to specify the format of the time information.**

**Each command in a pipeline is executed as a separate process (i.e., in a subshell).**

**awk '{ print $5 }' :- pattern scanning and processing language**

**program**

**If no -f option is specified, the first operand to awk shall be the text of the awk program. The application shall supply the program operand as a single argument to awk. If the text does not end in a <newline>, awk shall interpret the text as if it did.**

**argument**

**Either of the following two types of argument can be intermixed:**

**file**

**A pathname of a file that contains the input to be read, which is matched against the set of patterns in the program. If no file operands are specified, or if a file operand is '-' , the standard input shall be used.**

**assignment**

**An operand that begins with an underscore or alphabetic character from the portable character set (see the table in the Base Definitions volume of IEEE Std 1003.1-2001, Section 6.1, Portable Character Set), followed by a sequence of underscores, digits, and alphabetics from the portable character set, followed by the '=' character, shall specify a variable assignment rather than a pathname. The characters before the '=' represent the name of an awk variable; if that name is an awk reserved word (see Grammar ) the behavior is undefined. The characters following the equal sign shall be interpreted as if they appeared in the awk program preceded and followed by a double-quote ( ' )' character, as a STRING token (see Grammar ), except that if the last character is an unescaped backslash, it shall be interpreted as a literal backslash rather than as the first**

**character of the sequence "\"" . The variable shall be assigned the value of that STRING token and, if appropriate, shall be considered a numeric string (see Expressions in awk ), the variable shall also be assigned its numeric value. Each such variable assignment shall occur just prior to the processing of the following file, if any. Thus, an assignment before the first file argument shall be executed after the BEGIN actions (if any), while an assignment after the last file argument shall occur before the END actions (if any). If there are no file arguments, assignments shall be executed before processing the standard input.**

**sort -n :- sort lines of text files**

**-n compare according to string numerical value**

**tail -n 1 :- output the last part of files**

**-n, --lines=K output the last K lines, instead of the last 10; or use -n +K to output lines starting with the Kth**

**sed 's/%//' :- stream editor for filtering and transforming text**

**If no -e, --expression, -f, or --file option is given, then the first non-option argument is taken as the sed script to interpret. All remaining arguments are names of input files; if no input files are specified, then the standard input is read.**

**---------------------------------Assignment given on 06/11/22 in resources ----------------------------**

**Assignment 1 :- How to upload HTML web pages on Apache2 web server in EC-2 Instance? Please justify with step by step answers.**

**Step1: create ec2 machine of your choice of flavour.**

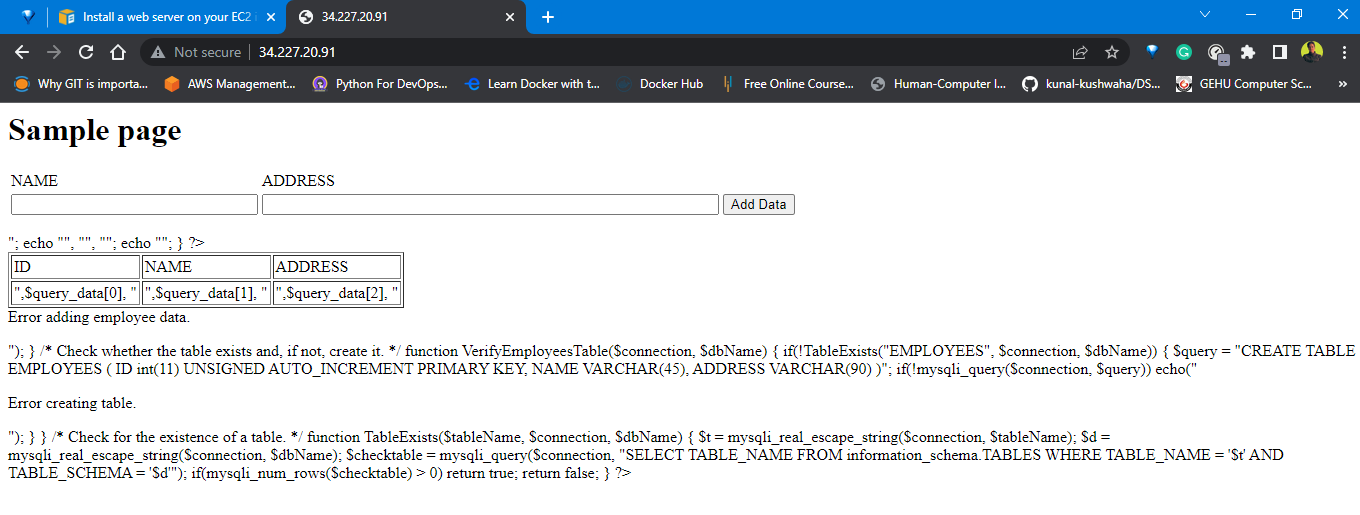
**Step2: update the machine (sudo yum update).**

**Step3: install ,start and update the httpd packgae (yum install httpd ,systemctl start httpd , systemctl enable httpd)**

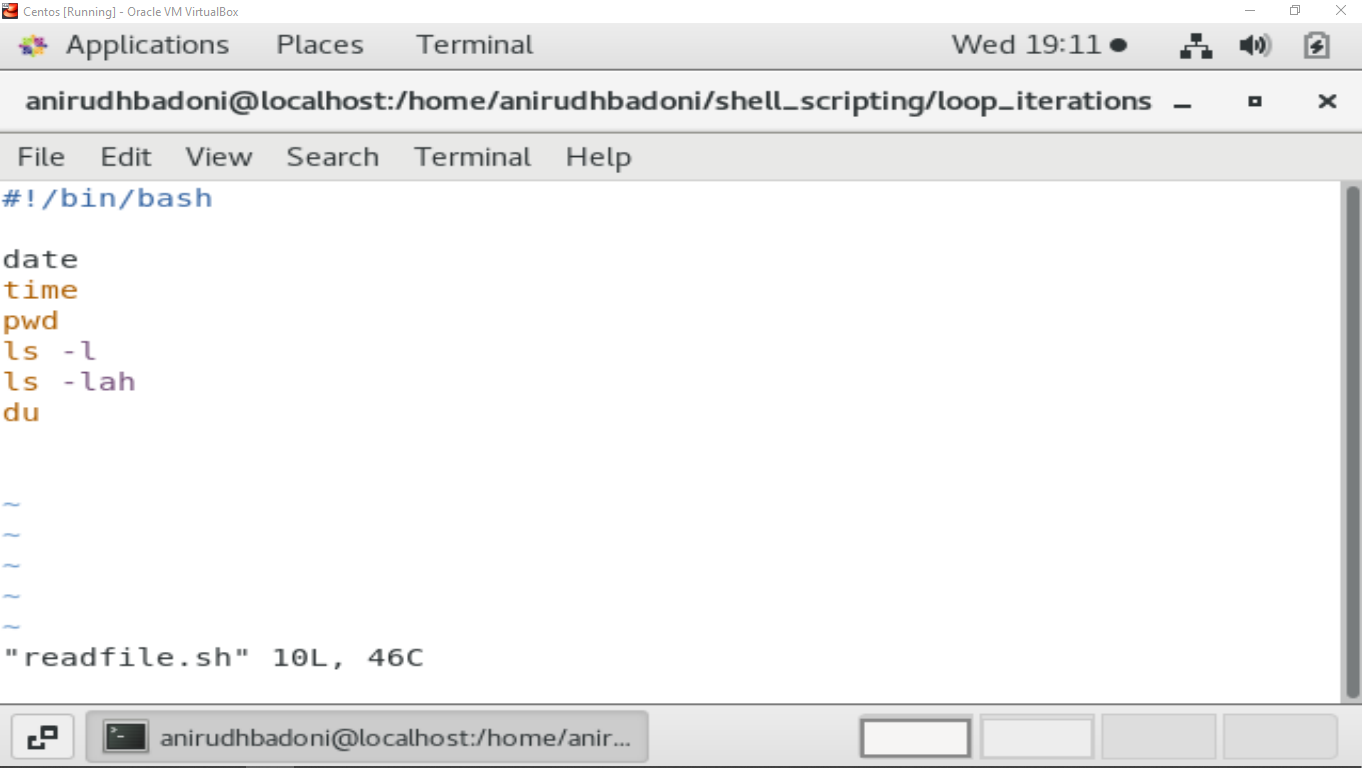
**Step4: change the inbound and outbound rule to http to run it on the browser.**

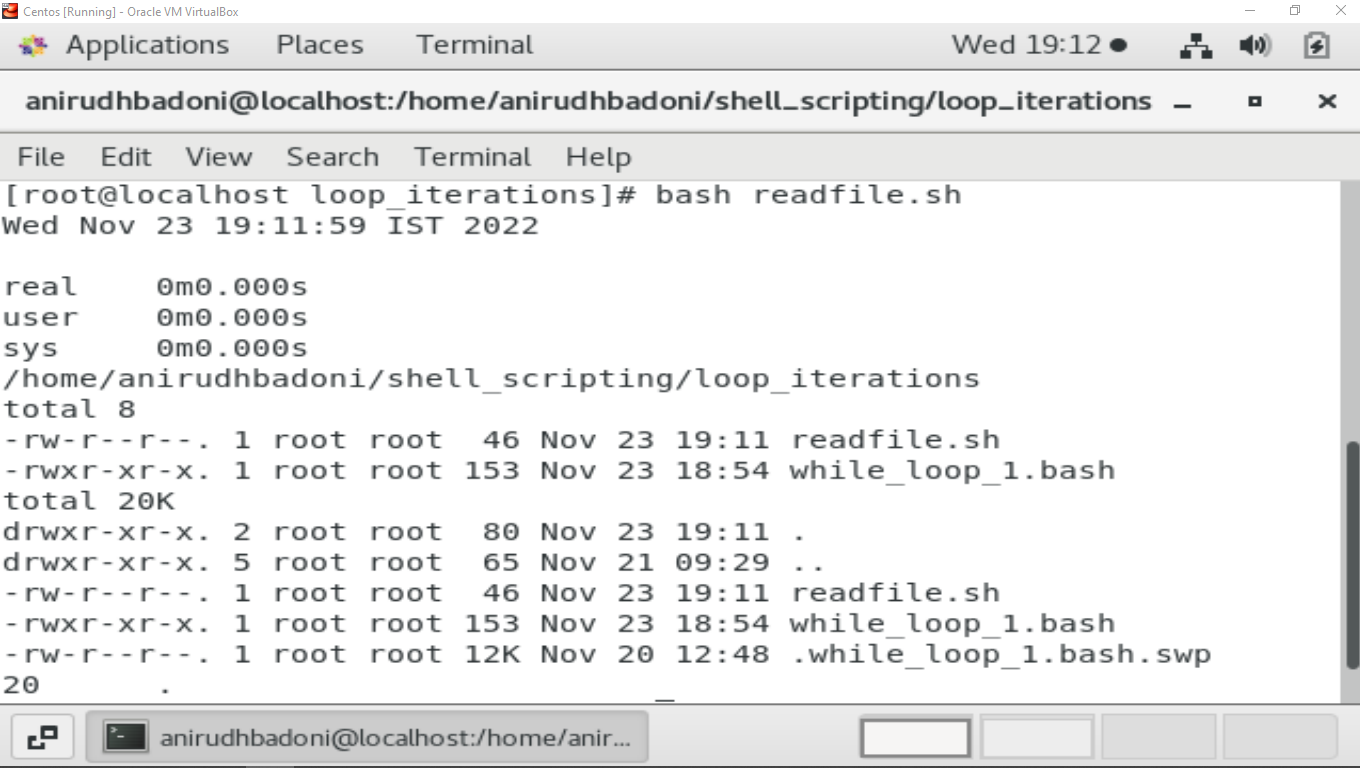
**Step5: got to var/www/html/ and create index.html file and add a html code of your choice.**

**Step6: paste the ip address of machine in the browser and click enter.**

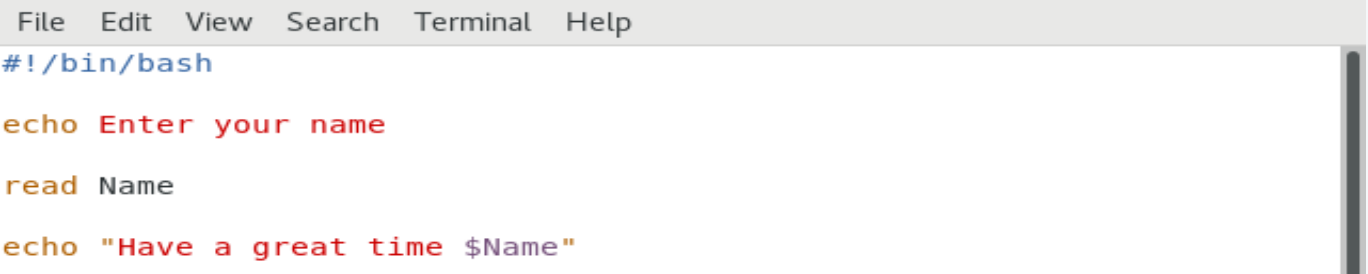
****

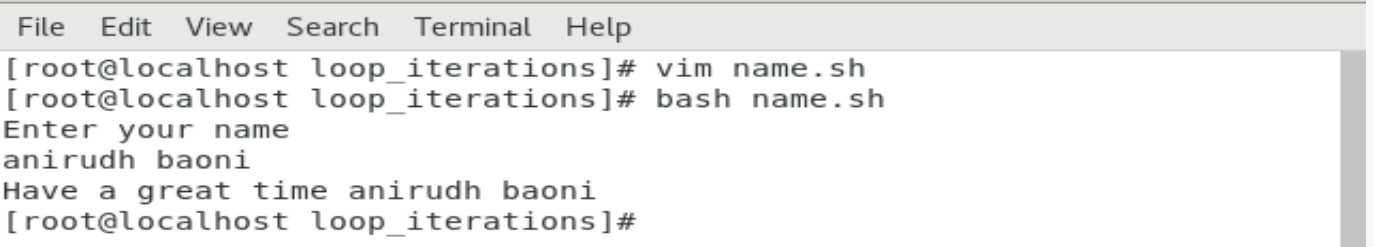
**Assignment 2 : Create readfile.sh in which you can read the information of PWD like size, permission, date time etc.**

****

****

**Assignment 3 : Take an input of name from user and print Have a great day ahead {name}**

****

****

**Assignment 4 : 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**

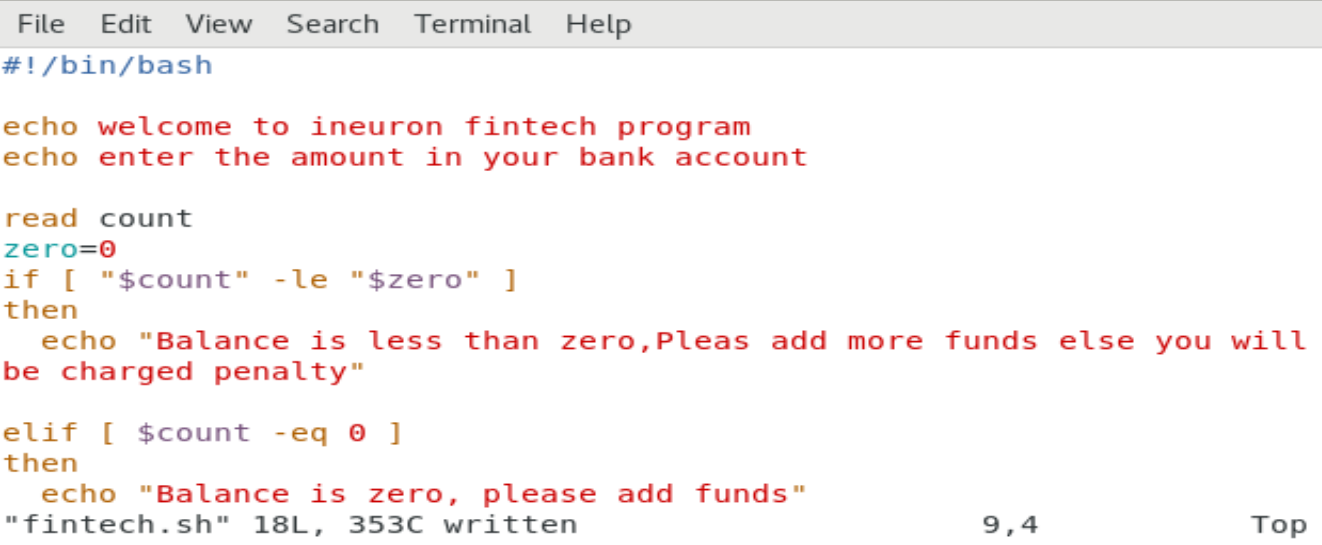
** The balance is above zero**

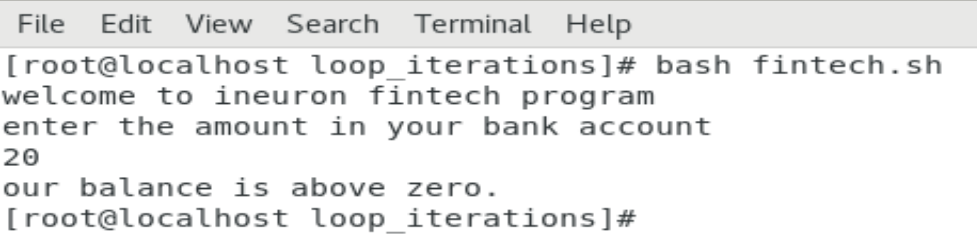
**For instance, in the following program, use the if, elif, else statements to display different outputs in different scenarios:**

**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.**

****

****