1. Write a shell script to display the specified number of lines from the given file. Accept a file name and a number of lines from the user during run time and perform the below mentioned validations:

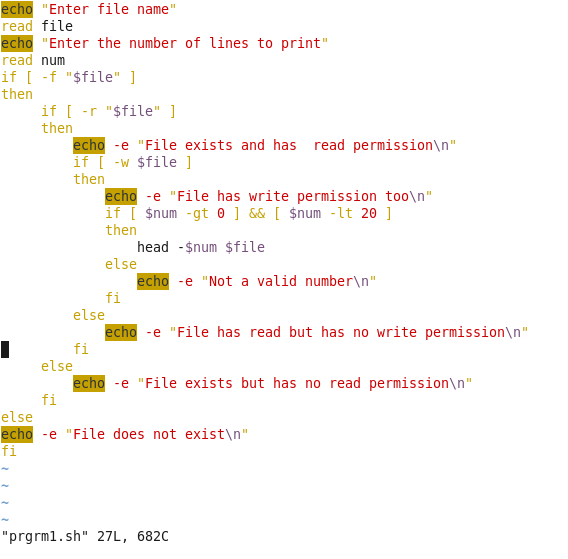
* 1. Check whether the file name entered exists and has both read and write

permission for the owner.

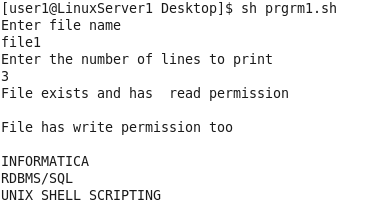
* 1. The number of lines should be > 0 and < 20.

If the above condition is true then display the specified number of lines from the given file name, otherwise display appropriate error message.

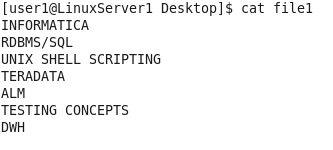
Program:



Output:



Data in file1:



2. Write a menu driven shell script to perform the following operations :-

* + 1. Make File.
    2. Display contents
    3. File Copy
    4. Rename the file
    5. Delete the file
    6. Exit

**1.** **Make File :** This module should accept the file name from the user and check for its existence and the size of the file. If it exist with a size > 0 then display an alert message “ File Already Exists “, Otherwise allow the user to create the file with some meaningful data into it. After performing the transaction prompt the user with the menu.

**2. Display Contents :** This module should accept a file name from the user. If the file exists, then display the contents of the file. If the file does not exist, then display suitable error message. After the process prompt the user with the menu display.

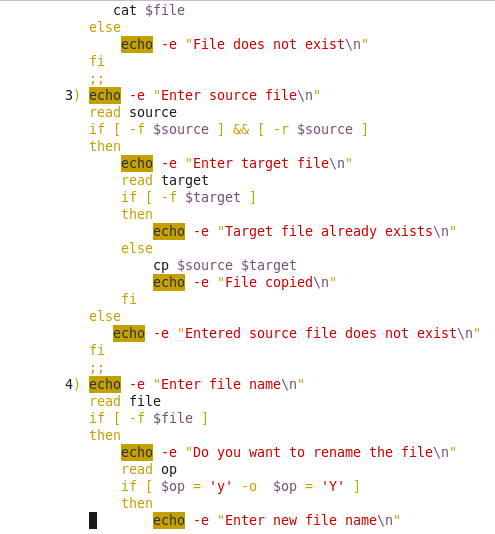
**3. File Copy :** This module should accept both source file and target file from the user. If the source file exists and is readable, then accept the target file name. If the source file does not exist, then display suitable error message. If the target file does not exist, then copy the contents of the source file to the target file. If the target file exists, then display suitable message and go back to the menu.

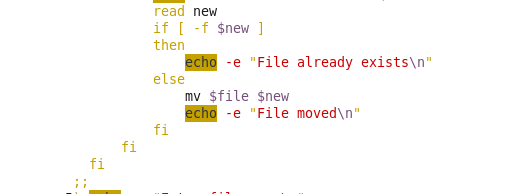
**4. Rename File :** This module should accept the name of the file from the user during run time. If it exists then prompt the user “ Do you want to Rename the file :” , if the user types “y” or “Y” then prompt the user to enter the destination file and check for its existence if it does not exists then rename the file along with the time stamp value. Otherwise display appropriate error message.

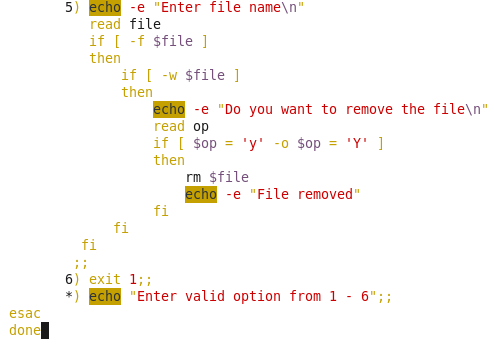
**5. Delete File :** Accepts the name of the file from the user. Check for the existence and the “write” permission for the file. If the file exists, then delete the file with confirmation from the user. If the file does not exist, then display suitable error message.

Program:

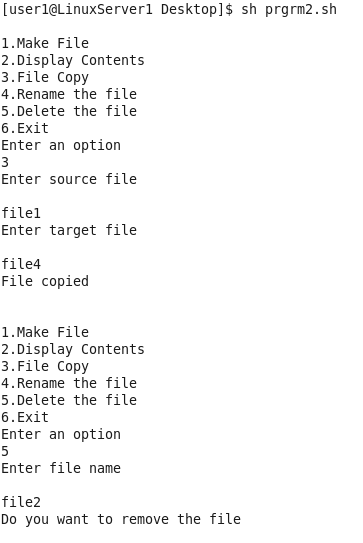


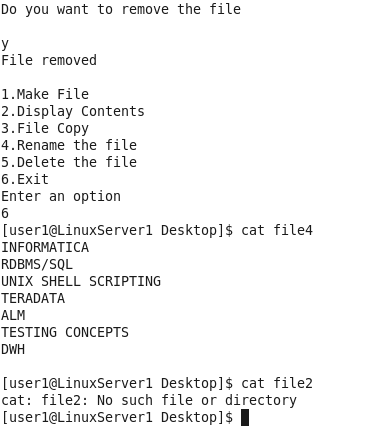


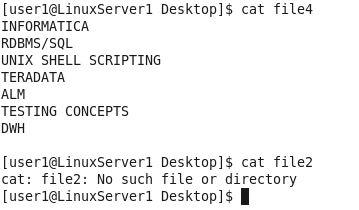


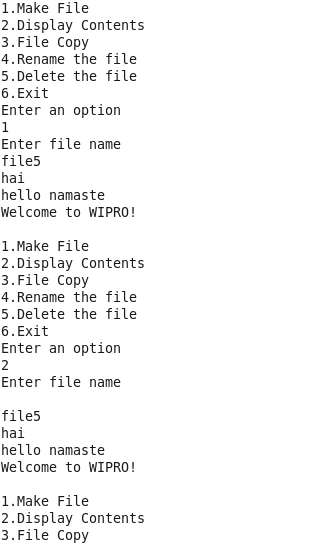


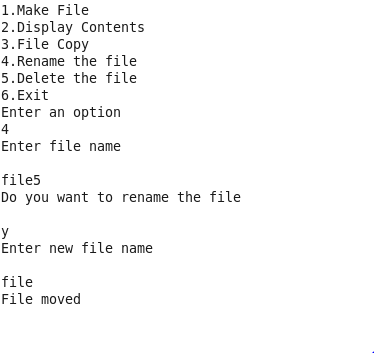
Output:

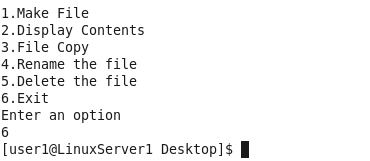












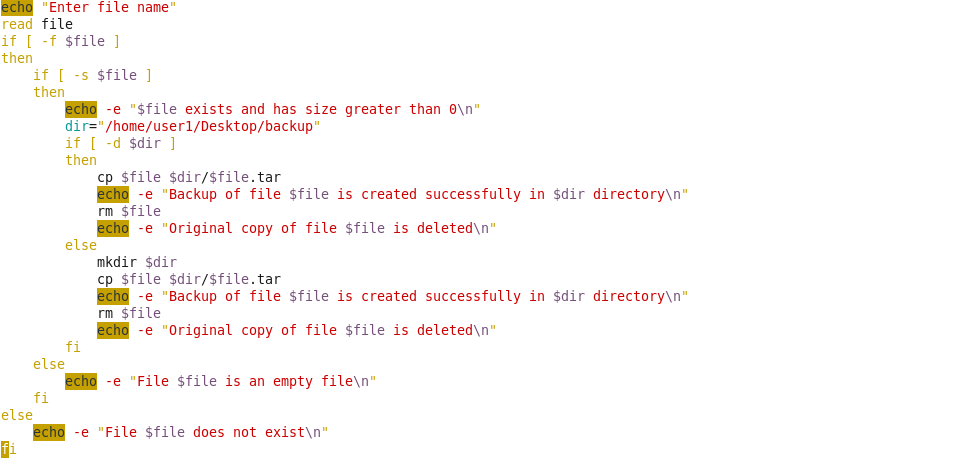
3. Write a shell script to create a backup for a file as per the below mentioned specifications:-

* 1. Accept the file name from the user during run time.
  2. Check for the existence of the file and the size should be > 0.
  3. If the above condition is true create a backup for the file as per below mentioned path.

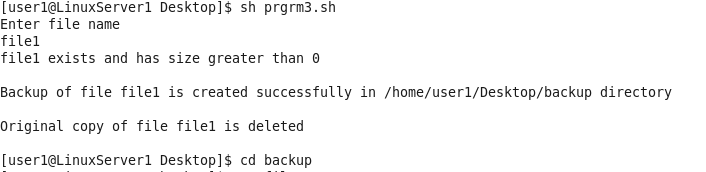
/home/user1/backup/<filename.tar>.

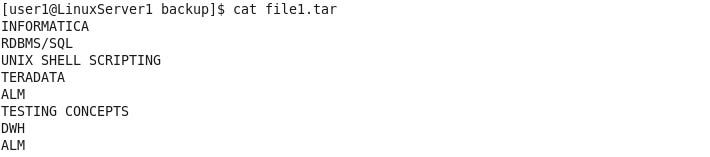
* 1. If the folder named “backup” does not exist then create the folder through the script and place the backups along with the time stamp value. Remove the original file.

Program:



Output:





4. Write a shell script to perform the following:-

a. Accept the filename and a string from the user during run time and perform the below mentioned validations.

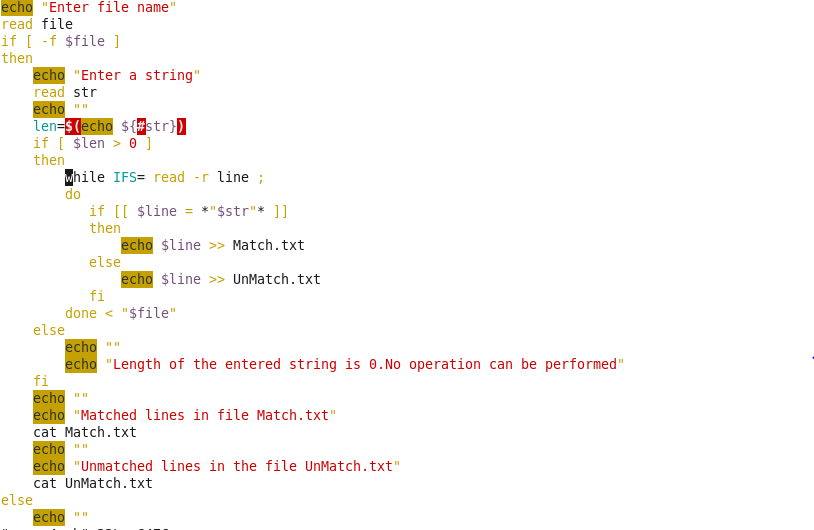
b. The length of the string should be > 0 and should contain only alphabets.

c. If the above condition is true check for the existence of the string in the given file name without using a “grep” command and redirect all the lines that matches with the entered string to a file named “Match.txt”. Otherwise the unmatched lines should be copied to a file named “UnMatch.txt”.

d. Display the number of lines, words and characters available in the file named Match.txt and UnMatch.txt with appropriate header message.

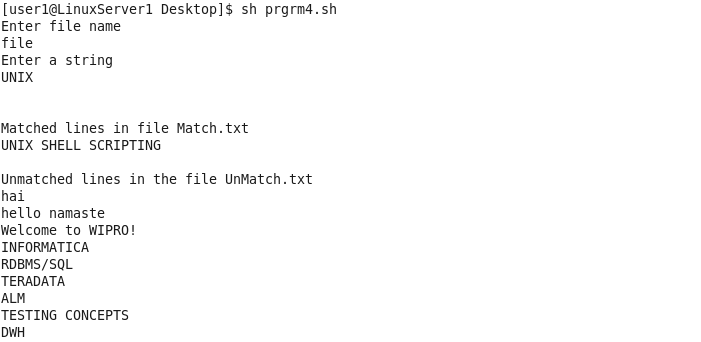
e. Display appropriate error messages wherever necessary.

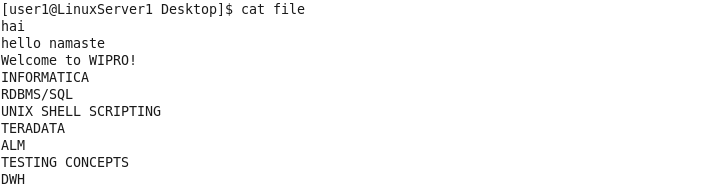
Program:





Output:

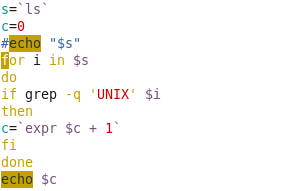




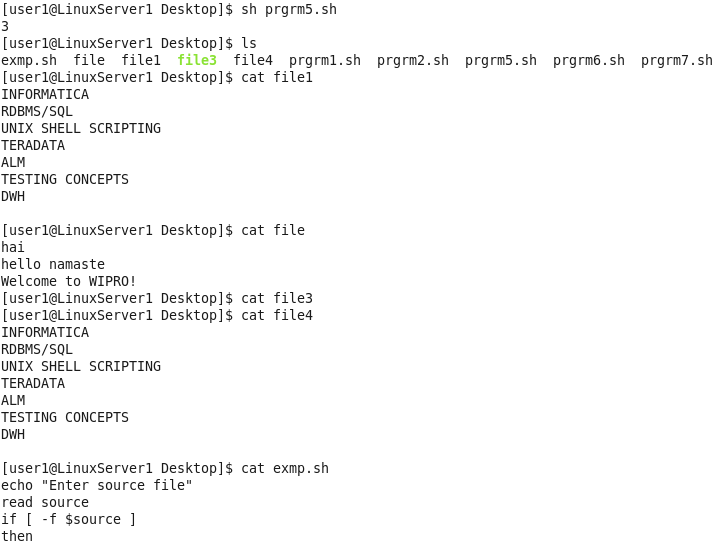
5. Write a shell script to count the total number of files in the current working directory containing the text “UNIX” in them.

Note: Prepare a few files with some text as well as the text “UNIX” in them to test it.

Program:



Output:

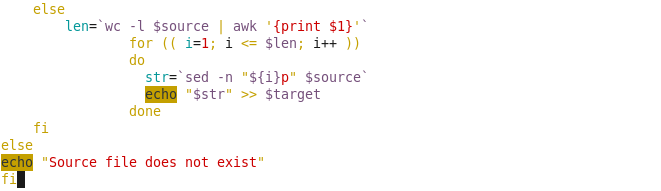


(The other file is the prgrm5.sh)

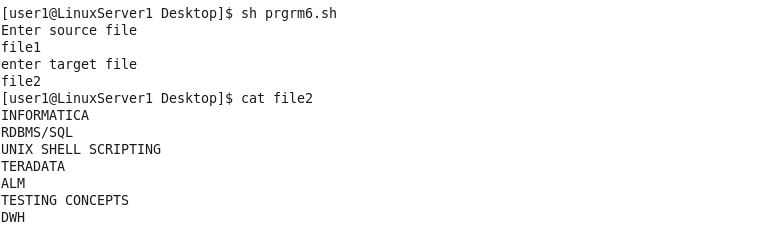
6. Write a shell script to copy a file without using a cp command and perform necessary validations.

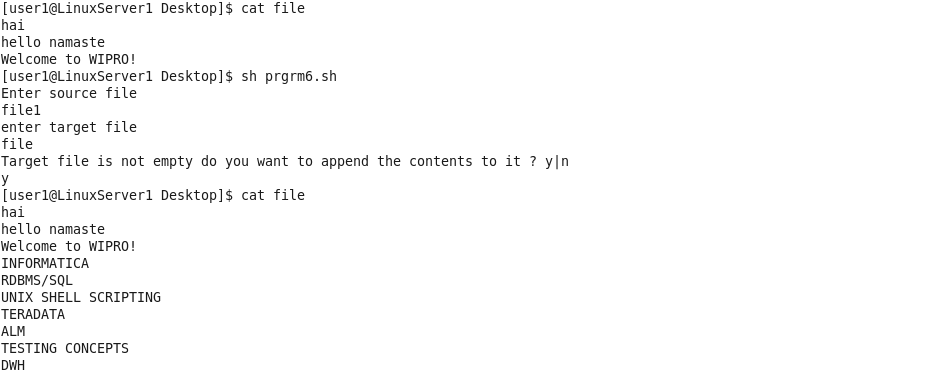
Program:

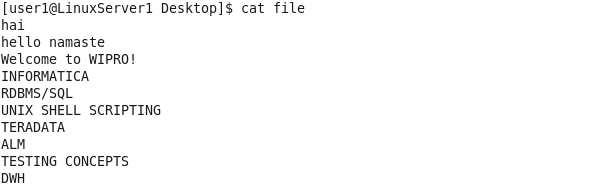




Output:

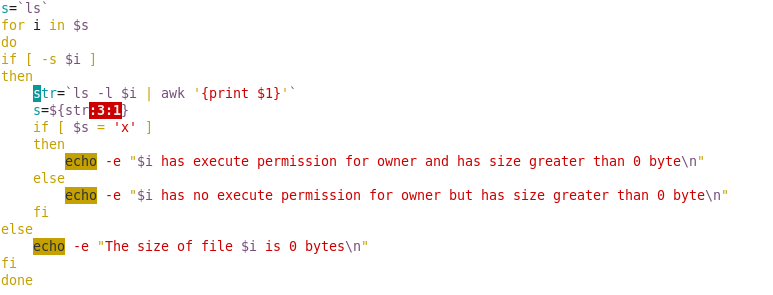




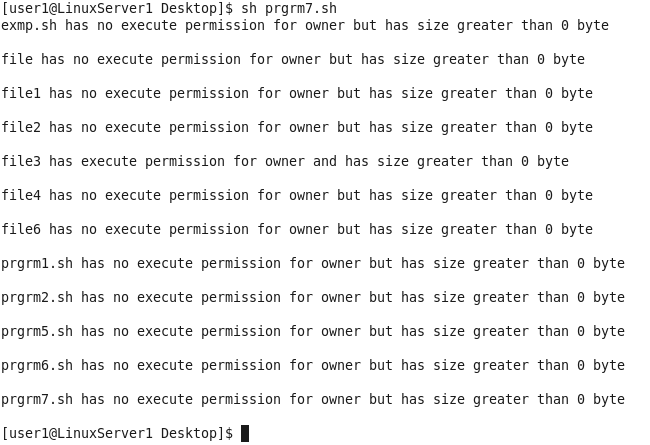


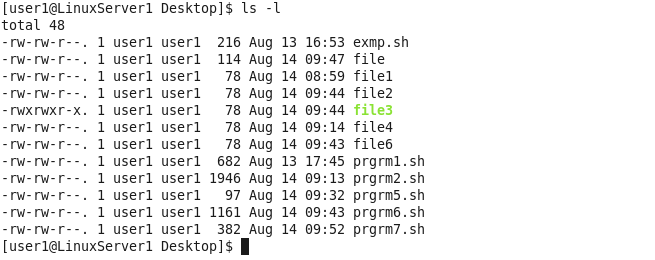
7. Write a shell script to display the ordinary files in the current working directory which has size > 0 and execute permission for the owner.

Program:



Output:

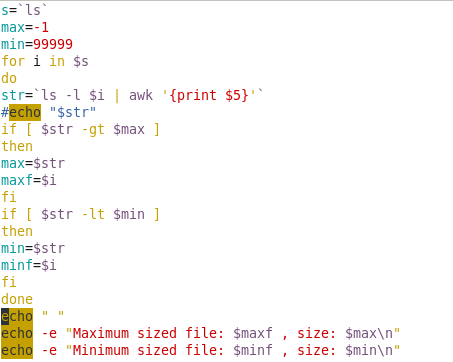




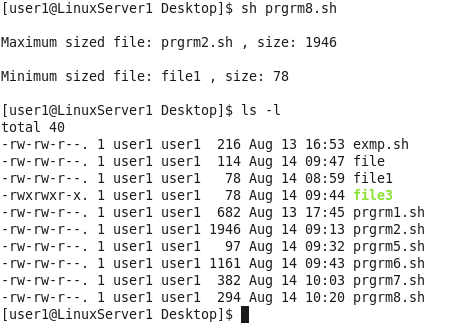
8. Write a shell script to display the maximum and minimum size of the files in the current working directory.

Note: Consider the size of the file in bytes.

Program:



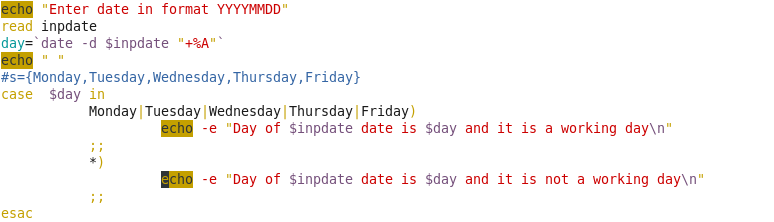
Output:



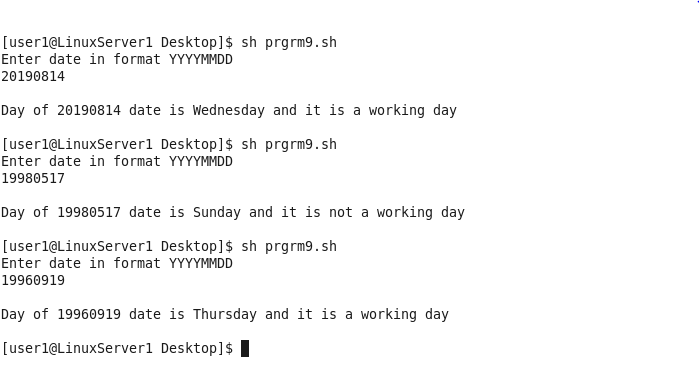
9. Write a shell script to accept a date from the user and check whether it is a working day or not?

Note: Working day should be between Mon-Fri.

Program:



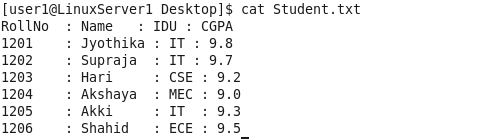
Output:



10. Write a shell script to update a student record as per the below mentioned specifications :-

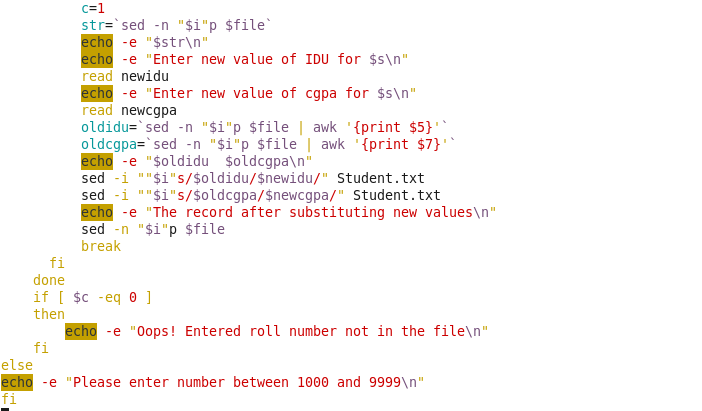
* 1. Consider the file named “Student.txt” exists with the following fields separated by “:” as a delimiter (RollNo, Name, IDU, CGPA).
  2. The script should accept the RollNo from the user during run time and perform the below mentioned validations.
  3. The RollNo should be a 4 digit positive integer ranging between 1000 to 9999. If the above condition is true check for the existence of the RollNo in the file named “Student.txt”.
  4. If it exist display the corresponding student record and prompt the user to change the IDU and CGPA and update the same in the file named “Student.txt”.
  5. Display appropriate error messages.

File: Student.txt

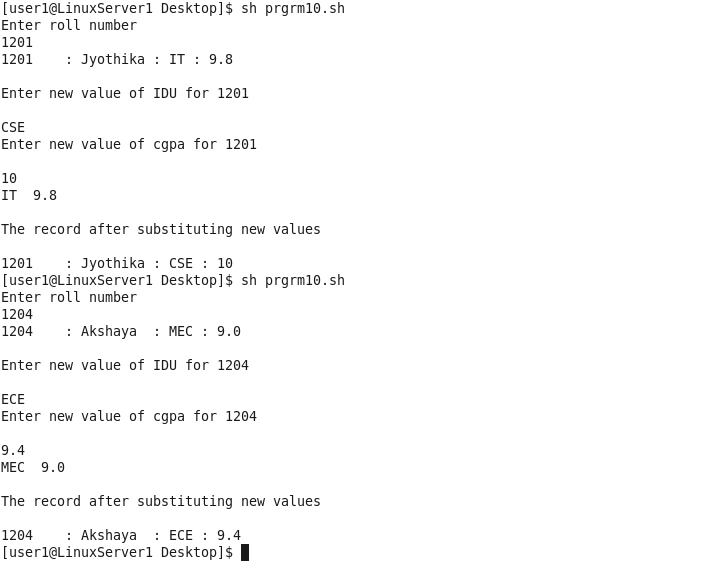


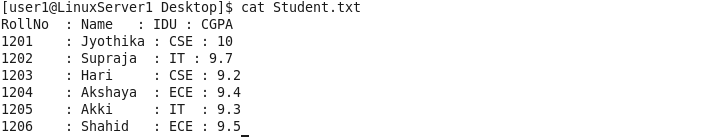
Program:





Output:





11. Write a shell script to perform the following operations

**a. SumofDigits :**  This function has to accept an integer number as a parameter and find the sum of the digits and print the same.

For Eg.: **I/p.: 128 O/p.: 2**

**b. ReverseOfNumber :**  This function has to accept an integer number as a parameter, reverse the given number, and print the reversed number.

For Eg.: **I/p.: 128 O/p.: 821**

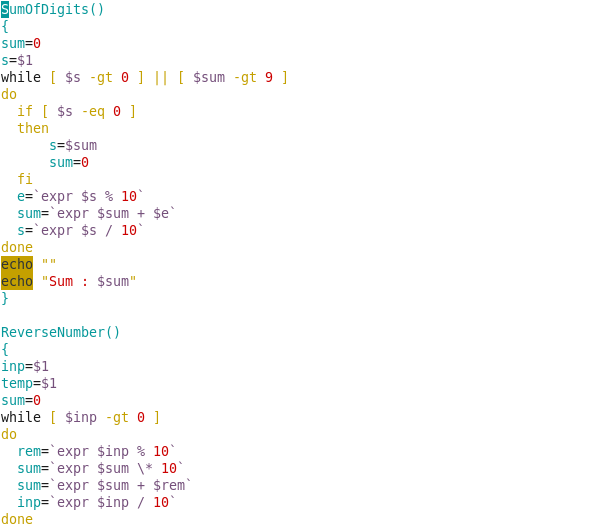
**c. ConvertBinary :** This function has to accept an integer number as a parameter and print the binary equivalent of the number.

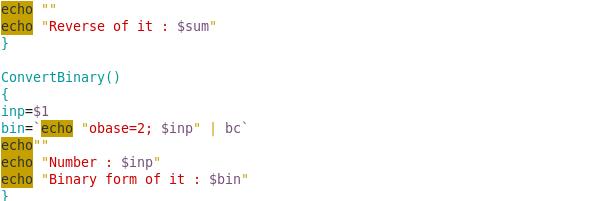
For Eg.: **I/p.: 10 O/p.: 1010**

Store all the functions in the file named **“Functions.sh”** and call all this functions in a file named **“Operations.sh”.**

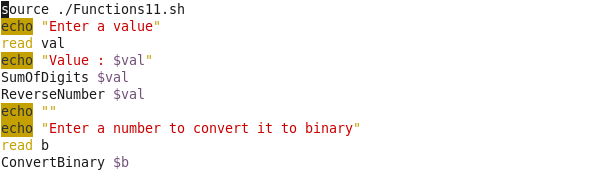
Program:

File: Functions11.sh

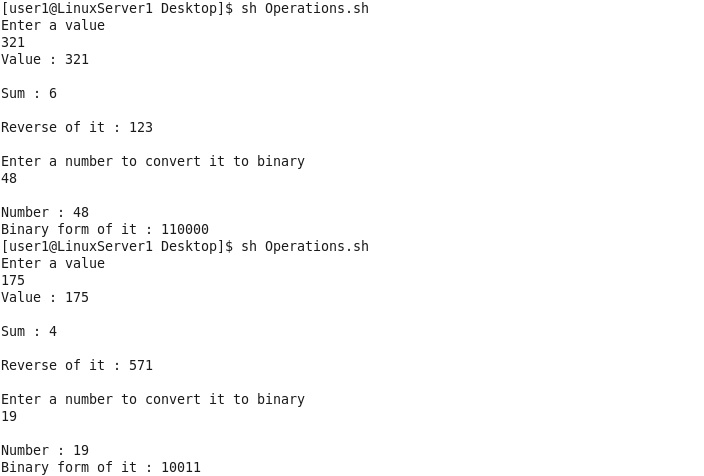




File: Operations.sh

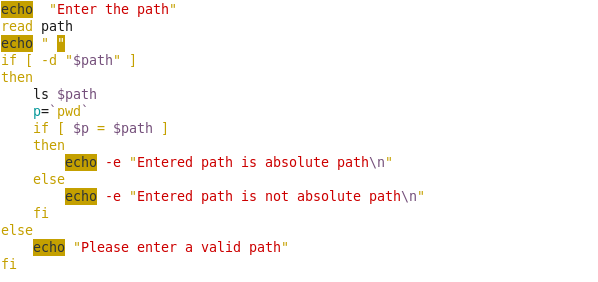


Output:

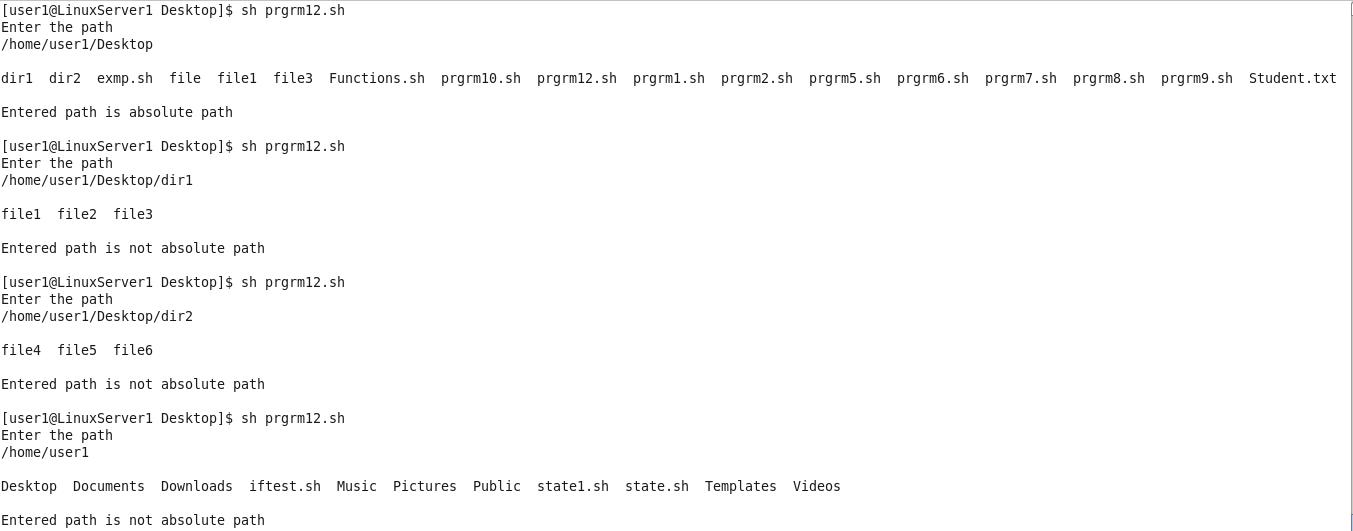


12. Write a shell script that accepts the path name as a parameter and check for the existence of the path, if it exist display all the files and sub-directories on that path as well as check whether the parameter passed is of either absolute or relative path, otherwise display appropriate error messages and do the necessary validations.

Program:



Output:





13. Predict the output of the following shell script :

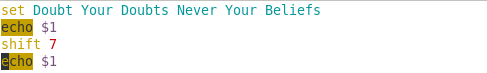
set Doubt Your Doubts Never Your Beliefs

echo $1

shift 7

echo $1

Program:



Output:



14. Write a shell script to generate the Payroll as per the below mentioned specifications from the file named **“EmpDetails.txt”** and display the report as per the format mentioned below.

**Contents of EmpDetails.txt**

**EmpName IDU Salary Additional Hours**

BMS TT 55000 30

Ajeesh ENR 40000 21

Raj HR 39000 18

Yuvi TT 37000 12

**I/p Specifications:**

1. Accept the IDU name as a command line argument to the awk script.
2. The argument passed should be a string.
3. Group all the employees who belong to that IDU and the sort the details of the employees based on the additional hours and display the report in the below mentioned format.

-----------------------------------------------------------------------------

Employee Wages Report

IDU Name:

------------------------------------------------------------------------------

S.No. EmpName Salary Overtime TotalWages

-- ---- --- ---- ----

-- ---- --- ---- ----

-- ---- --- ---- ----

------------------------------------------------------------------------------

Total Amount Disbursed:

------------------------------------------------------------------------------

**Note :**

a. Calculate the wages as per the following guidelines:

Salary <=10000, Additional Per Hour Rs. 150

Salary >10000 & <=20000, Additional Per Hour Rs. 200

Salary >20000 & <=30000, Additional Per Hour Rs. 250

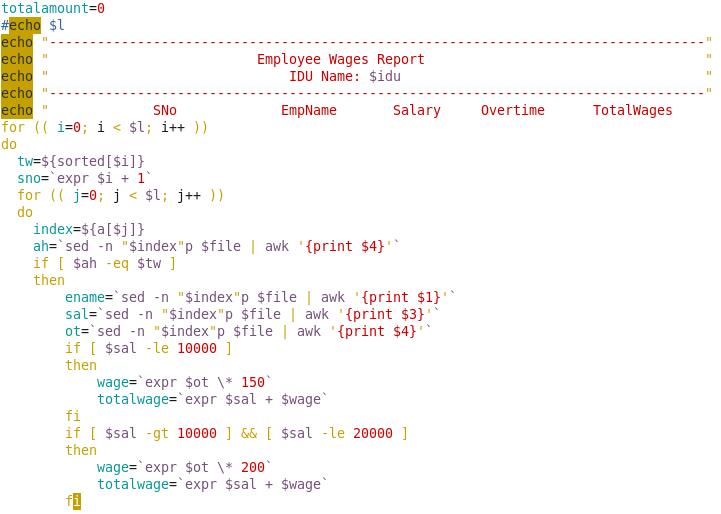
Salary >30000 Additional Per Hour Rs. 300

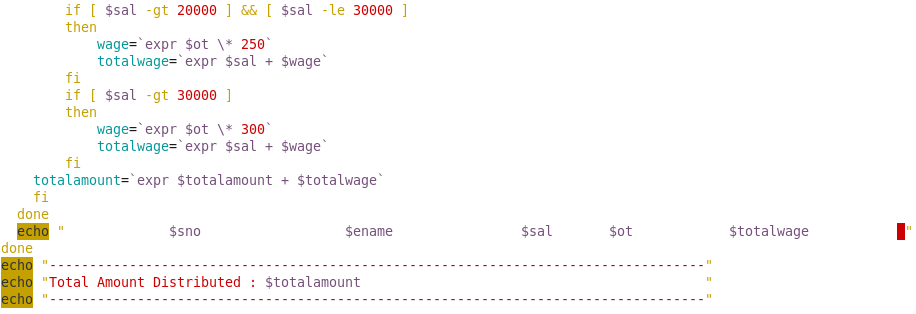
**Total Wages = Salary + Overtime \* <Value at Step a>**

b. **Total Amount Disbursed = sum of all the Total Wages**

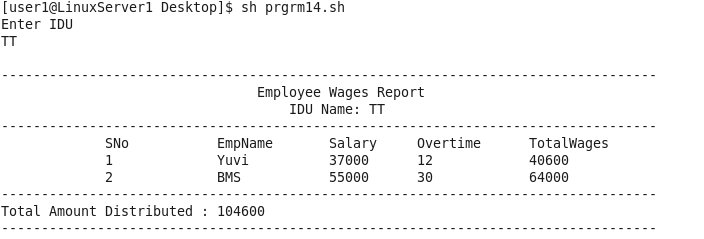
Program:

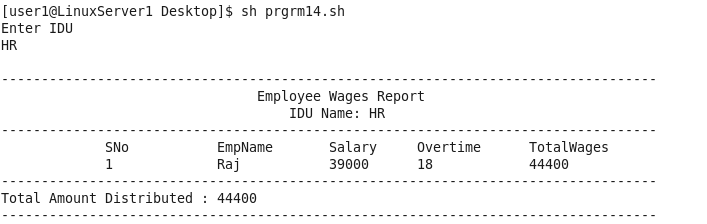


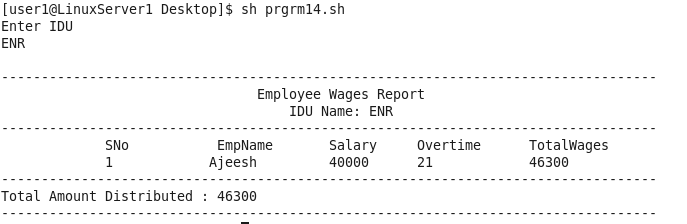




Output:







15. Write a shell script to generate the Course wise report as per the below mentioned specifications from the file named **“Courses.txt”** , “**Faculty.txt”** and display the report as per the format mentioned below.

**Contents of Courses.txt**

**Class Subject Facultyid Results**

BCA UNIX 4080 98%

BCA UNIX 3090 100%

MCA CD 4080 85%

MCA UNIX 4080 98%

**Contents of Faculty.txt**

**FacultyId FacultyName YearOfExperience**

3090 Ajeesh 10

4080 Magesh 10

4071 Raj 12

4062 Yuvi 9

**I/p Specifications :**

* 1. Pass Subject as a command line argument to the awk script.
  2. Check for the number of command line arguments and the argument should be string.
  3. Generate the report for the above mentioned subject based on the faculty who has given the results between 80 to 100 as per the below mentioned format.

**Performance Report**

**Subject :**

**-------------------------------------------------------------------------------------**

**Faculty Id Faculty Name Class Result**

**---- ------ --- ---**

**---- ------ --- ---**

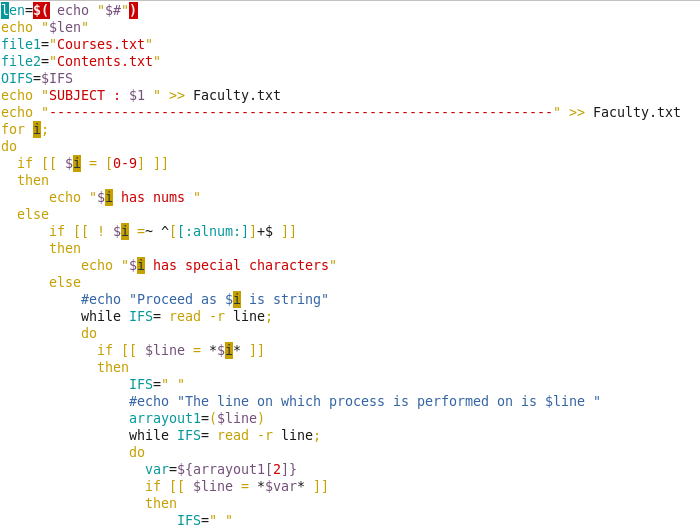
**---- ------ --- ---**

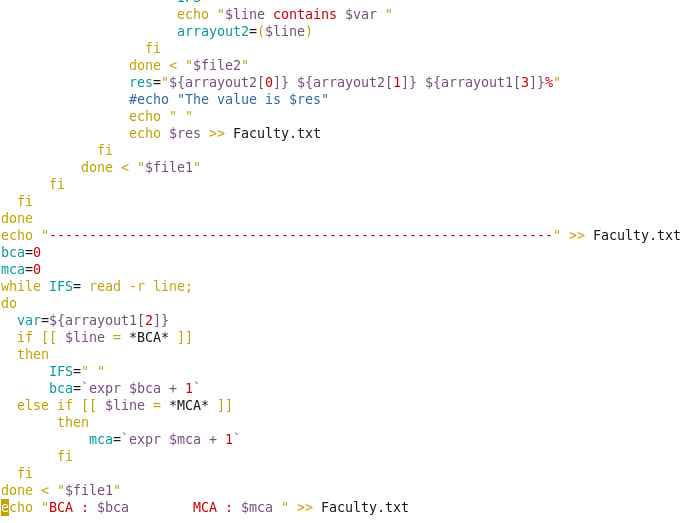
**-------------------------------------------------------------------------------------**

**BCA : <No of Faculty> MCA : <No of Faculty>**

* 1. Finally display total number of faculties who can handle the given subject as a footer message.

Program:





Output:









16. Create a library file **“Functions.sh”** for the following functions:-

a. **fnValidate() :** This function will check whether  the passed parameter is a 3 digit positive number. If yes return 1 and If not return 0.

b. **fnCheck() :** This function check whether the passed parameter iseven or odd.

c. **fnConvert() :** This function check the passed parameter digit by digit. If digit is a even number then that digit should be replaced by 1 for that many occurrences and if it’s odd then that digit should be replaced by 0 for that many occurrences.

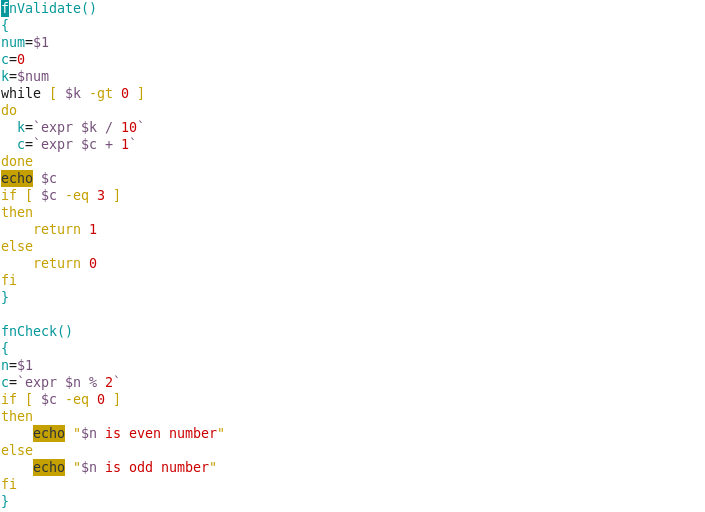
**For Eg.: I/p. : 234 O/p. : 110001111**

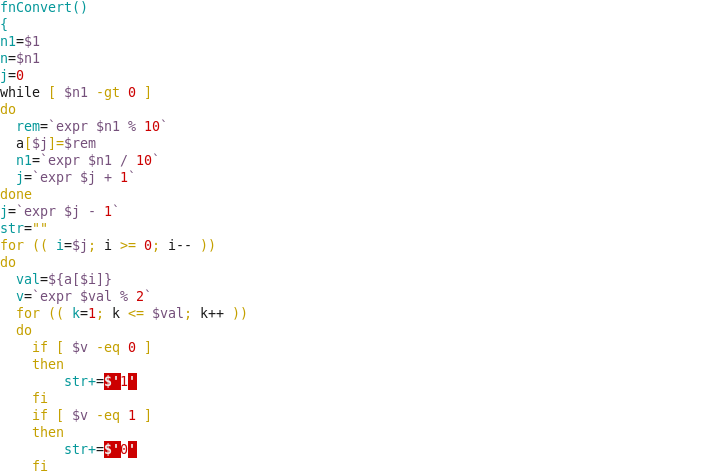
**Note :**  The function **fnCheck()** **and fnConvert()** has to be invoked if and only if the return value of function **fnValidate()** is equal to 1 otherwise display appropriate error message.

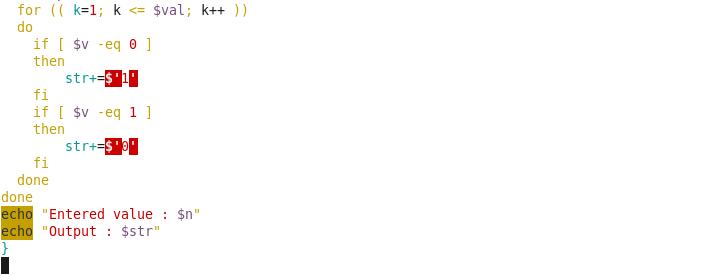
Call all the above functions from a shell script **main.sh** by passing a parameter to it.

Program:

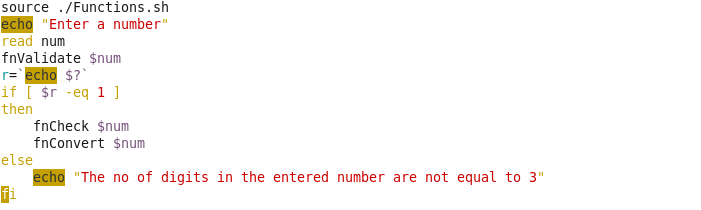
File: Functions.sh



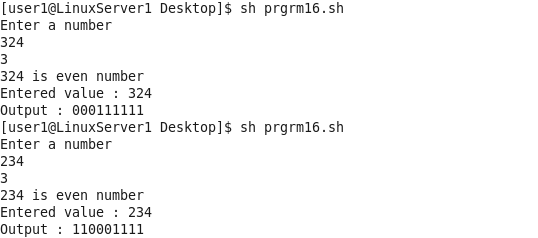


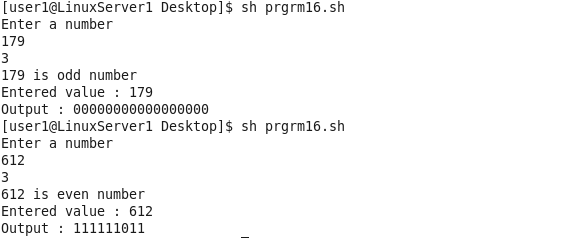


File: prgrm16.sh



Output:





17. Create a library file **“Functions.sh”** for the following functions:-

**a. fnValidate() :** Pass all the command line argument as a parameter to the function, check whether the parameter passed is an integer. If the above condition is satisfied return 1 otherwise return 0.

**b. fnMaxMin() :** This function should accept all the command line arguments as a parameter and display the maximum and minimum numbers among the given set of numbers.

**c. fnSumOfDigits() :** This function should accept all the command line arguments as a parameter and find the sum of the digits of all the parameters as mentioned below.

**For Eg.: I/p.: 10 20 -30 48 O/p.: 10 + 20 – 30 + 48**

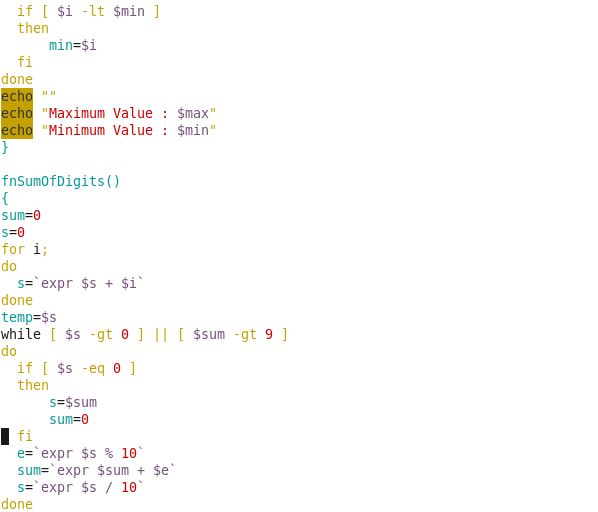
**48**  **4 + 8 12 3**

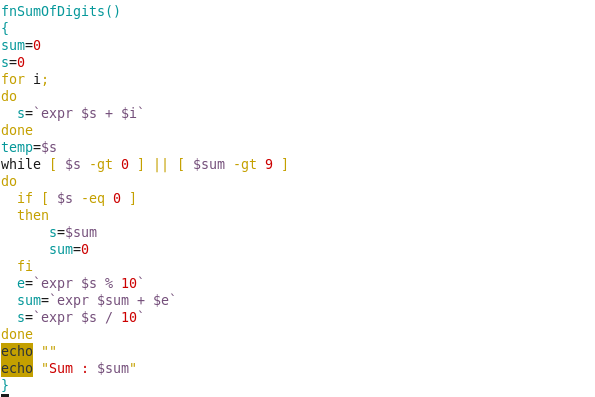
Call all the above functions from a shell script **Manipulations.sh** by passing 4 parameters.

Program:

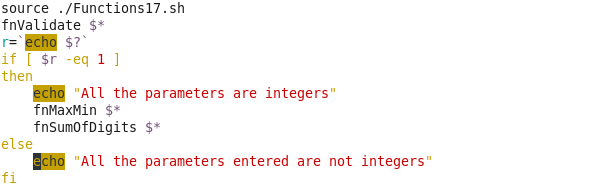
File: Functions17.sh



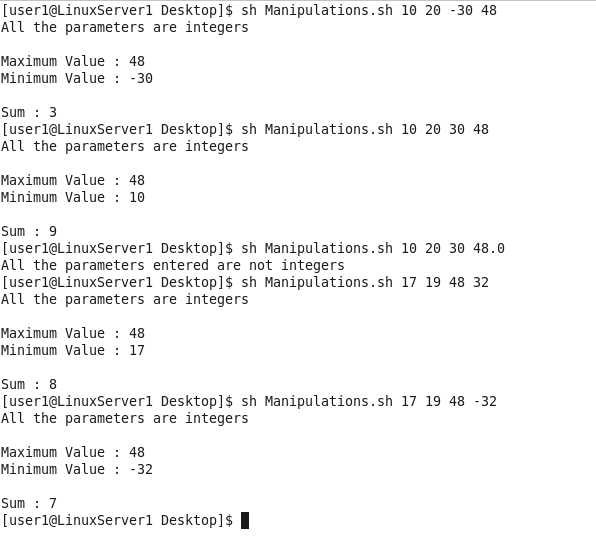




File: Manipulations.sh



Output:



18. Write a shell script to perform the following operations using functions

**a. String Compare :** This function has to accept two different strings and check whether the given strings are equal or not.

**b. Display Position :** This function has to accept two parameters both of string datatype, In which the first one is used to have the string and the second one is known as the searching pattern, Search the pattern on the given string and display in which position the searching pattern is available.

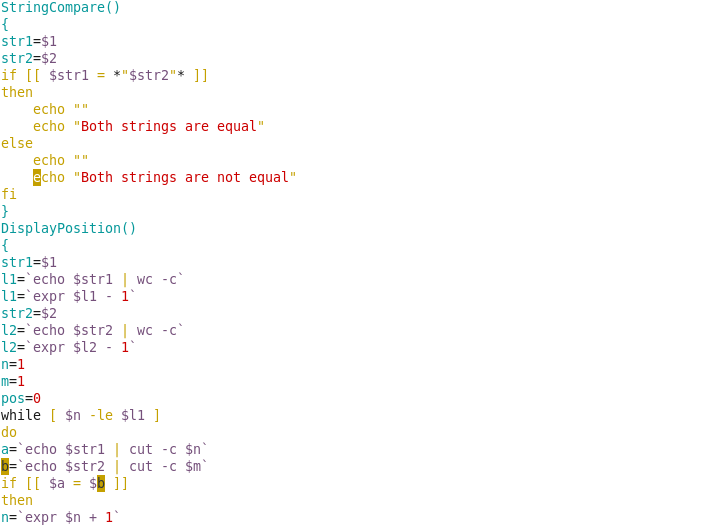
For Eg : **I/p : Doubt Your Doubts Never Your Beliefs, Doubts**

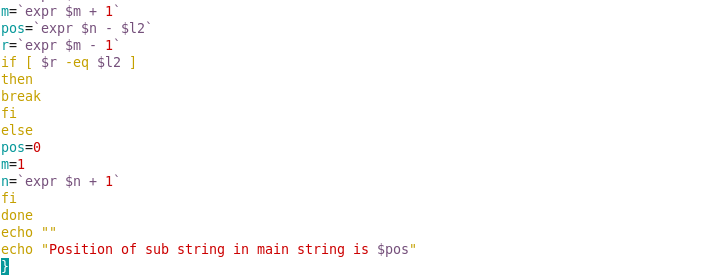
**O/p : 12**

Store all the functions in the file named **“Strings.sh”** and call all this functions in a file named **“Mainpulate.sh”** and perform the below mentioned validations for the parameters before passed to the function (**i.e., Strings should not be null).**

Program:

File: Strings.sh

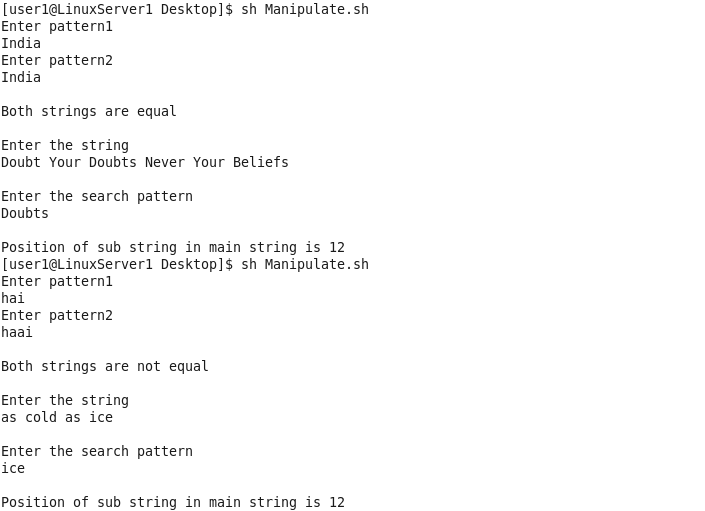


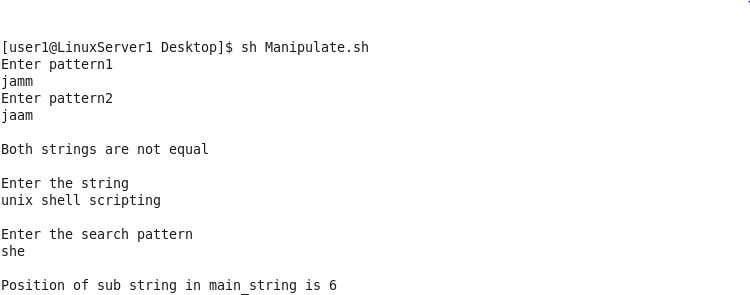


File: Manipulate.sh



Output:





20. Write a shell script to do the following operations using functions

**a. String Occurrences :** It is used to accept a 2 Strings as a parameter and check the total number of occurrences of the second string in first String.

**(Note: Without using grep command).**

**For Eg.:**  **I/P.: String1 :** Doubt Your Doubts Never Your Beliefs

**String2 :** Your

**O/p:** 2

**b. Counting Number of Words :**  It is used to accept the string as a parameter and count the total number of words in the given string.

**(Note : Without using wc command)**

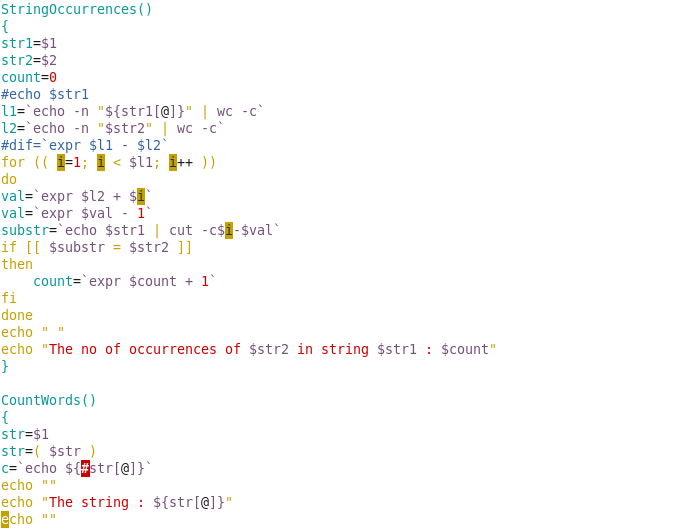
**For Eg. : I/p.:** Doubt Your Doubts Never Your Beliefs

**O/p.:** 6

Place these users defined functions in a file named “Functions.sh” and call these functions in a file named **“String.sh”**.

Program:

File: Functions20.sh

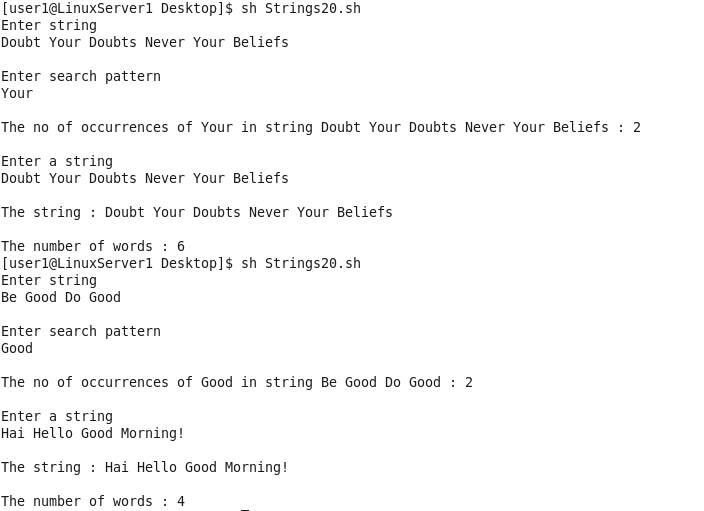




File: Strings20.sh



Output:



29. Write a shell script to perform the following operations using functions

**a. fnWordCount() :** This function should accept a string as a parameter

and it should display the total number of words available in that string .

**[ Note.: without using wc command. ] For Eg.: Doubt Your Doubts Never Your Beliefs**

**O/p.: 6**

**b. fnStringNConcat() :** This function should accept 3 arguments as a parameter 2 should be string and the 3rd argument should be an positive integer and concatenate that many number of characters from the second string to the first string and store the result in the first string.

**For Eg.: I/p: String1 : Wipro String2: Technologies NoOfCharacters : 3**

**O/p.: WiproTec**

**c. fnCharacterCount() :** This function should accept a string as an argument and it should display the number of repeated characters in that string.

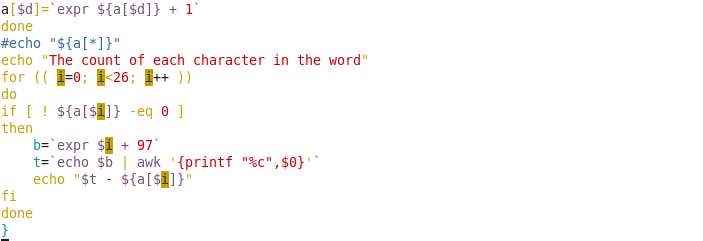
**For Eg.: I/p.: madam O/p.: m – 2 a - 2 d - 1**

Store all the functions in the file named **“Manipulations.sh”** and call all this functions in a file named **“FileMain.sh”.**

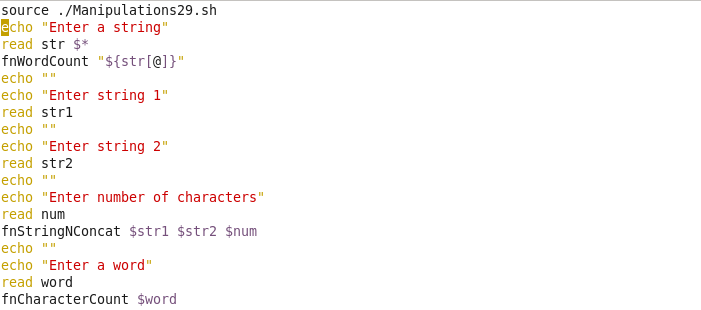
Program:

File: Manipulations29.sh





File: FileMain.sh



Output:

