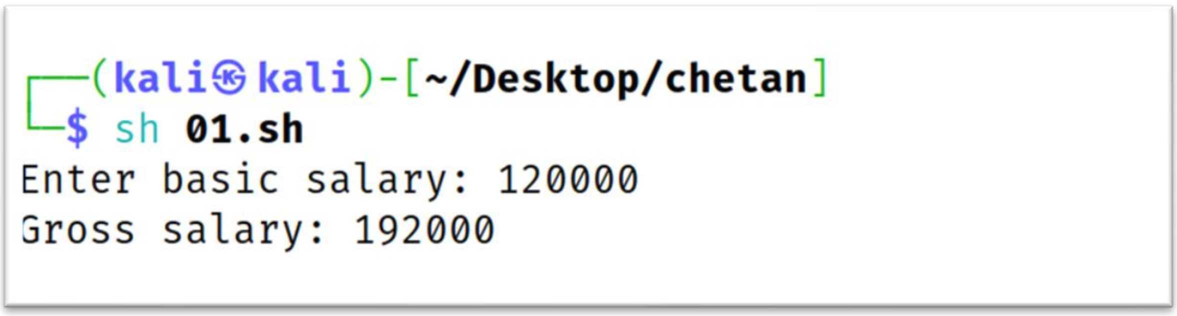
Problem Sheet-2

Sub:- Linux Fundamentals

1. The person' basic salary is input through keyboard. dearness allowance is 40% of the basic salary and HRA is 20% of the basic salary calculate the gross salary.

```
# Prompt user for basic salary
echo -n "Enter basic salary: "
read basic_salary
# Calculate dearness allowance and HRA
da=$((basic_salary * 40 / 100))
hra=$((basic_salary * 20 / 100))
# Calculate gross salary
gross_salary=$((basic_salary + da + hra))
# Print result
echo "Gross salary: $gross_salary"
```

OUTPUT:-

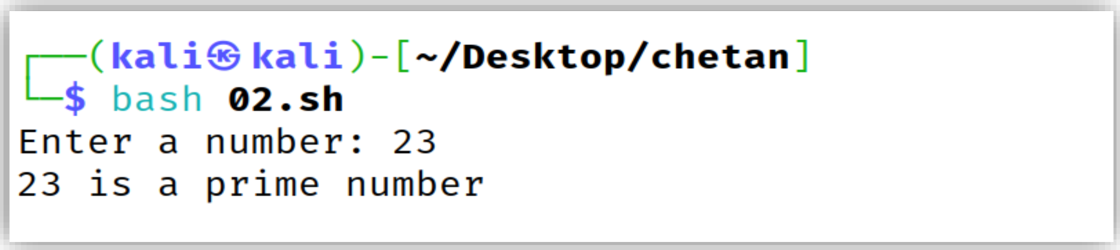


```
┌──(kali㉿kali)-[~/Desktop/chetan]
└─$ sh 01.sh
Enter basic salary: 120000
Gross salary: 192000
```

2. Write a script to check whether the given number entered from is prime or not.

```
echo -n "Enter a number: "
read number
# Check if the number is prime
is_prime=1
for (( i=2; i<=$((number-1)); i++ ))
do
  if [ $((number%i)) -eq 0 ]
  then
    is_prime=0
    break
  fi
done
if [ $is_prime -eq 1 ]
then
  echo "$number is a prime number"
else
  echo "$number is not a prime number"
fi
```

OUTPUT:-

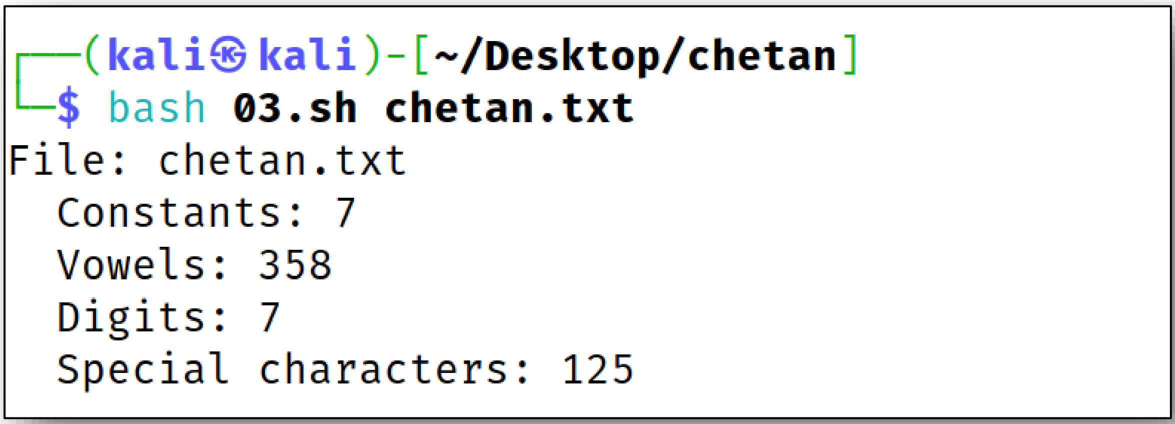


```
┌──(kali㉿kali)-[~/Desktop/chetan]
└─$ bash 02.sh
Enter a number: 23
23 is a prime number
```

3. Write a script that receives any number of filenames as argument and then count number of constants, vowels, digits, and special characters in each file

```
# Iterate over each file specified as an argument
for file in "$@"
do
  # Initialize counters for constants, vowels, digits, and special
characters
  constants=0
  vowels=0
  digits=0
  special_chars=0

  # Read each line of the file
  while read line
  do
    # Iterate over each character in the line
    for (( i=0; i<${#line}; i++ ))
    do
      # Check if the character is a constant, vowel, digit, or special
character
      # and increment the corresponding counter
      case ${line:i:1} in
        [A-Z] ) constants=$((constants+1)) ;;
        [a-z] ) vowels=$((vowels+1)) ;;
        [0-9] ) digits=$((digits+1)) ;;
* ) special_chars=$((special_chars+1)) ;;
     esac
   done
  done < $file

  # Print results for the file
  echo "File: $file"
  echo "  Constants: $constants"
  echo "  Vowels: $vowels"
  echo "  Digits: $digits"
  echo "  Special characters: $special_chars"
done
```

OUTPUT:-

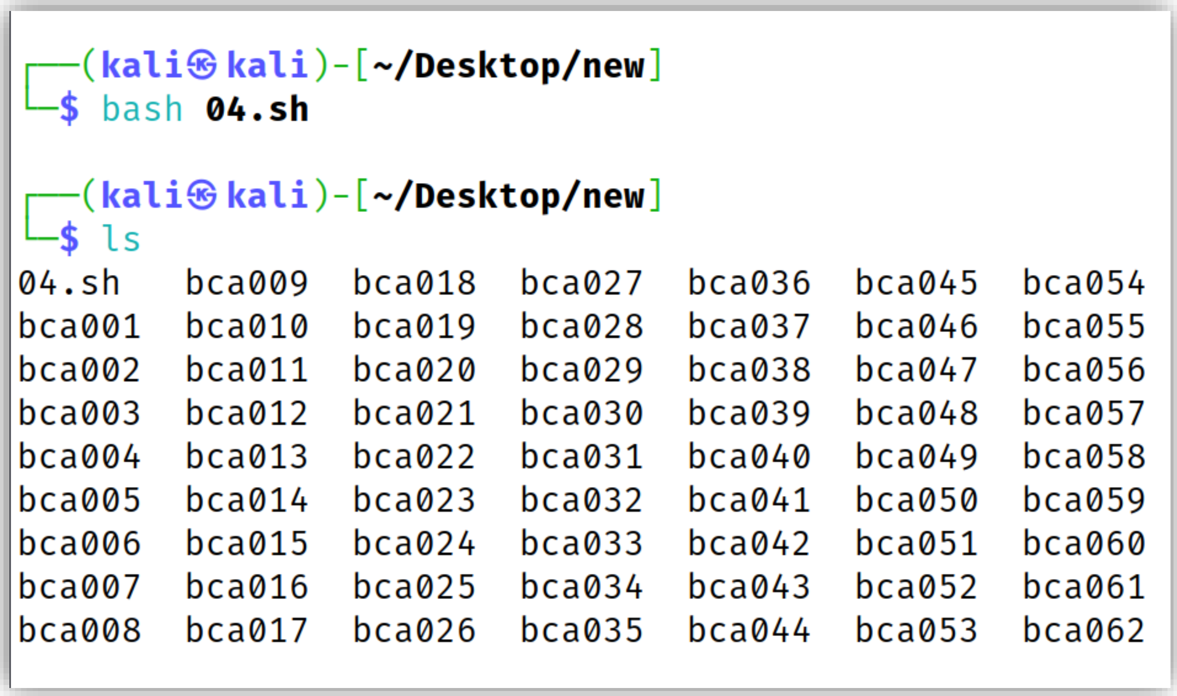


```
┌──(kali㉿kali)-[~/Desktop/chetan]
└─$ bash 03.sh chetan.txt
File: chetan.txt
  Constants: 7
  Vowels: 358
  Digits: 7
  Special characters: 125
```

4. Write a script that creates 100 files with the name bca001 up to bca100.

```
#!/bin/bash

# Create a for loop to iterate 100 times
for i in {1..100}
do
    # Use printf to generate the file name with leading zeros
    filename=$(printf "bca%03d" $i)

    # Create an empty file with the generated filename
    touch $filename
done
```

OUTPUT:-

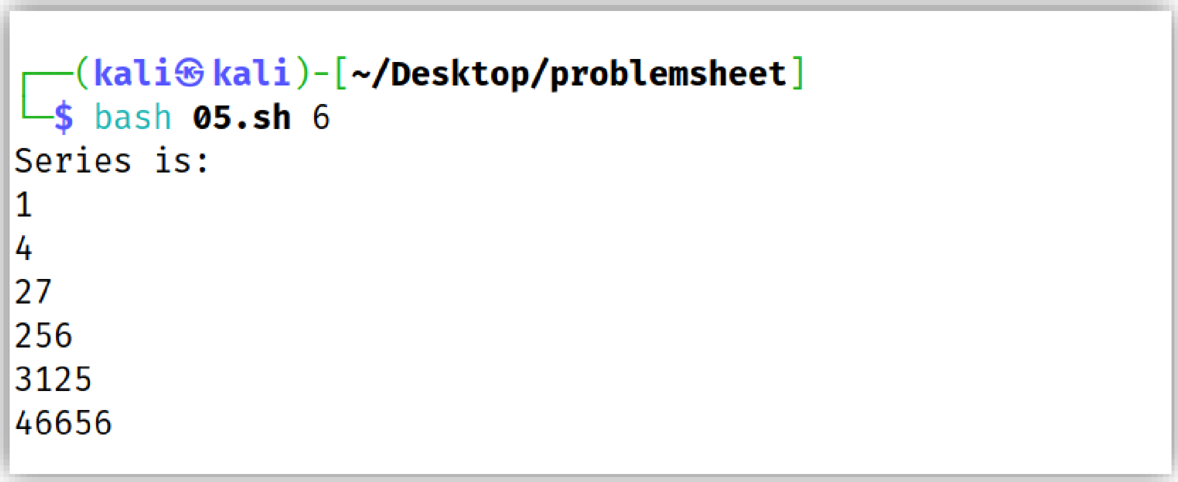


```
┌──(kali㉿kali)-[~/Desktop/new]
└─$ bash 04.sh

┌──(kali㉿kali)-[~/Desktop/new]
└─$ ls
04.sh   bca009  bca018  bca027  bca036  bca045  bca054
bca001  bca010  bca019  bca028  bca037  bca046  bca055
bca002  bca011  bca020  bca029  bca038  bca047  bca056
bca003  bca012  bca021  bca030  bca039  bca048  bca057
bca004  bca013  bca022  bca031  bca040  bca049  bca058
bca005  bca014  bca023  bca032  bca041  bca050  bca059
bca006  bca015  bca024  bca033  bca042  bca051  bca060
bca007  bca016  bca025  bca034  bca043  bca052  bca061
bca008  bca017  bca026  bca035  bca044  bca053  bca062
```

5. Write shell script to print following series: 1,4 ,27, 256,….

```
echo "Series is:"

# Loop from 1 to 9 (inclusive)
for i in $(seq 1 $1); do
 c=1

 # Loop from 1 to the current value of i
 for j in $(seq 1 $i); do
  # Multiply the current value of c by the current value of i
  c=$((c * i))
 done
 # Print the result
 echo $c
done
```

OUTPUT:-



```
┌──(kali㉿kali)-[~/Desktop/problemsheet]
└─$ bash 05.sh 6
Series is:
1
4
27
256
3125
46656
```

6. Write a script to make the following file and management operation menu based.

1. Display the current directory. 2. List directory.

3. make directory. 4. Change directory.

5. Copy of a file 6. Rename a file.

7. Delete a file. 8. Edit a file

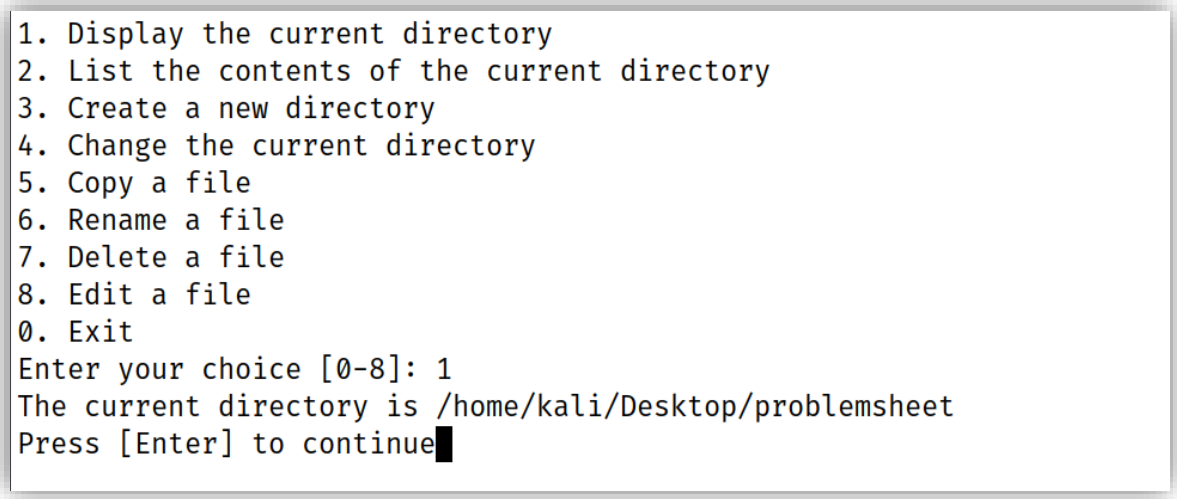
```
# Function to display the current directory
current_dir() {
  # Print the current directory to the screen
  echo "The current directory is" $(pwd)
}
# Function to list the contents of the current directory
list_dir() {
  # List the contents of the current directory
  ls
}


# Function to create a new directory
make_dir() {
  read -p "Enter the name of the new directory: " dir
  mkdir $dir
}
```

```
# Function to change the current directory
change_dir() {
  read -p "Enter the path of the directory to change to: " dir
  cd $dir
}

# Function to copy a file
copy_file() {
  read -p "Enter the path of the file to copy: " file
  read -p "Enter the path of the destination: " destination
  cp $file $destination
}

# Function to rename a file
rename_file() {
  read -p "Enter the path of the file to rename: " file
  read -p "Enter the new name for the file: " new_name
  mv $file $new_name
}

# Function to delete a file
delete_file() {
  read -p "Enter the path of the file to delete: " file
  rm $file
}
```

```
# Function to edit a file
edit_file() {
  read -p "Enter the path of the file to edit: " file
  vi $file
}

# Menu
while :
do
  # Clear the screen
  clear

  # Display menu
  echo "1. Display the current directory"
  echo "2. List the contents of the current directory"
  echo "3. Create a new directory"
  echo "4. Change the current directory"
  echo "5. Copy a file"
  echo "6. Rename a file"
  echo "7. Delete a file"
  echo "8. Edit a file"
  echo "0. Exit"

  # Read the user's input
```

```
    read -p "Enter your choice [0-8]: " choice

    # Check the user's input and call the appropriate function
    case $choice in
      1) current_dir;;
      2) list_dir;;
      3) make_dir;;
      4) change_dir;;
      5) copy_file;;
      6) rename_file;;
      7) delete_file;;
      8) edit_file;;
      9) exit;;
      esac
      read -p "Press [Enter] to continue"
  done
```

OUTPUT:-

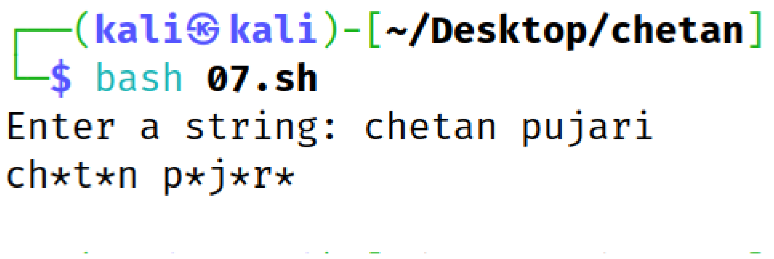


```
1. Display the current directory
2. List the contents of the current directory
3. Create a new directory
4. Change the current directory
5. Copy a file
6. Rename a file
7. Delete a file
8. Edit a file
0. Exit
Enter your choice [0-8]: 1
The current directory is /home/kali/Desktop/problemsheet
Press [Enter] to continue
```

7. Write shell script to replace all vowel with *.

```
#!/bin/bash

# Prompt user for a string
echo -n "Enter a string: "
read string

# Replace all vowels in the string with *
new_string=$(echo "$string" | sed -e 's/[aeiouAEIOU]/*/g')

# Print the resulting string
echo "$new_string"
```

OUTPUT:-



```
┌──(kali㉿kali)-[~/Desktop/chetan]
└─$ bash 07.sh
Enter a string: chetan pujari
ch*t*n p*j*r*
```

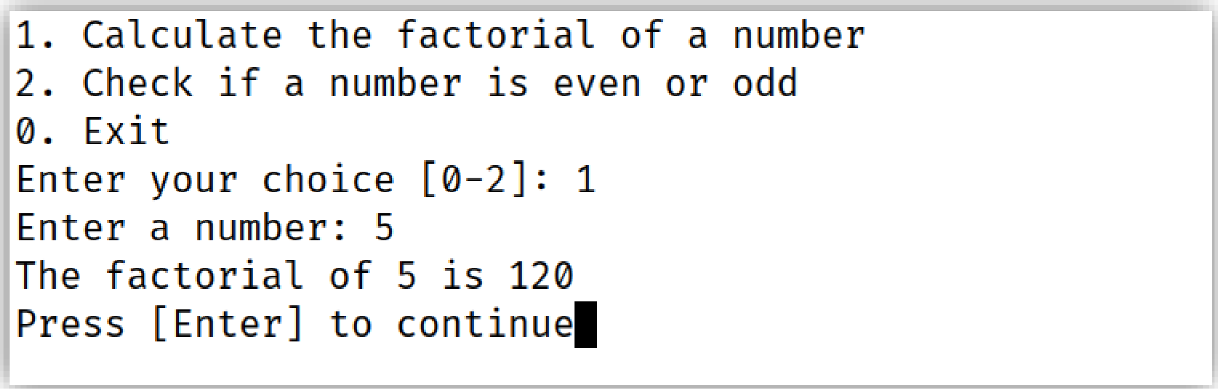
8. Write menu driven script which perform:

i) Find factorial of given number.

ii) Check whether given number is even or odd.

```
# Function to calculate the factorial of a number
factorial() {
  read -p "Enter a number: " num
  result=1
  # Calculate the factorial of the number
  for ((i=1; i<=num; i++))
  do
    result=$((result * i))
  done
  # Print the result to the screen
  echo "The factorial of $num is $result"
}
# Function to check if a number is even or odd
even_odd() {
  read -p "Enter a number: " num
  # Check if the number is even or odd
  if [[ $((num % 2)) -eq 0 ]]
  then
    echo "$num is an even number"
  else
    echo "$num is an odd number"
  fi
}
while :
do
  clear
  echo "1. Calculate the factorial of a number"
  echo "2. Check if a number is even or odd"
  echo "0. Exit"
  # Read the user's input
  read -p "Enter your choice [0-2]: " choice
  # Check the user's input and call the appropriate function
  case $choice in
    1) factorial;;
    2) even_odd;;
    0) exit 0;;
    *) echo "Invalid choice";;
  esac
  read -p "Press [Enter] to continue"
done
```

OUTPUT:-

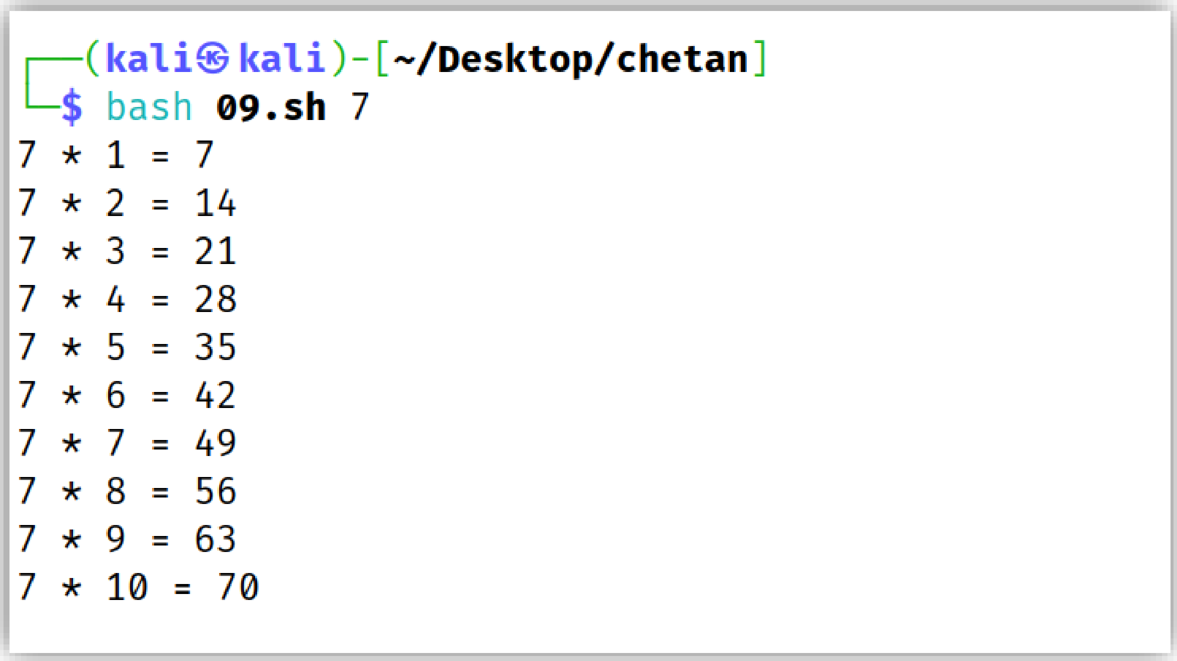


```
1. Calculate the factorial of a number
2. Check if a number is even or odd
0. Exit
Enter your choice [0-2]: 1
Enter a number: 5
The factorial of 5 is 120
Press [Enter] to continue
```

9. Write shell script to generate multiplication table for given number which passed as argument on cmd.

```
# Get the number to generate the multiplication table for
number=$1


# Use the "seq" command to generate a sequence of numbers from 1 to 10
for i in $(seq 1 10); do
  # Use the "expr" command to perform the multiplication and print the result
  result=$(expr $i \* $number)
  echo "$number * $i = $result"
done
```

OUTPUT:-

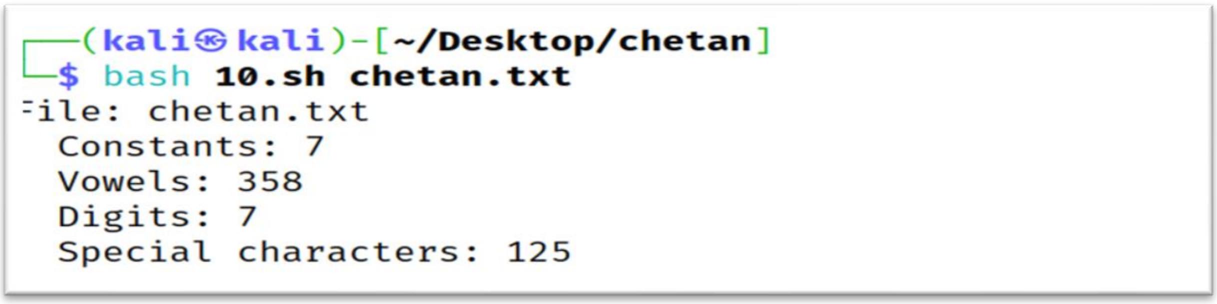


```
┌──(kali㉿kali)-[~/Desktop/chetan]
└─$ bash 09.sh 7
7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
7 * 7 = 49
7 * 8 = 56
7 * 9 = 63
7 * 10 = 70
```

10. Write a script that received any number of file as argument and then count number of vowel, digit and special character.

SAME AS QUESTION 7

OUTPUT:-



```
┌──(kali㉿kali)-[~/Desktop/chetan]
└─$ bash 10.sh chetan.txt
File: chetan.txt
  Constants: 7
  Vowels: 358
  Digits: 7
  Special characters: 125
```

11. Write menu-driven script for the following option.

1. Convert decimal binary number

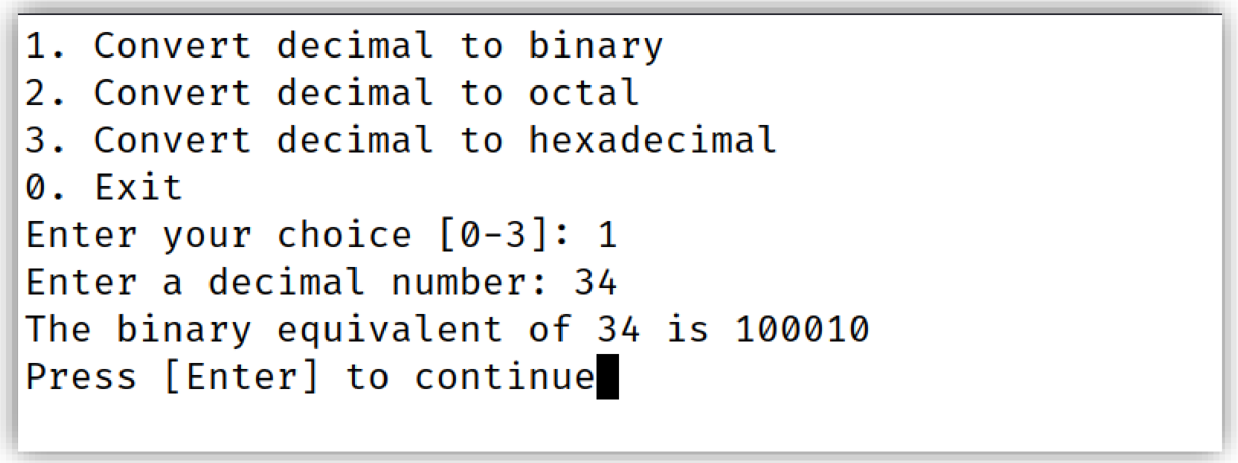
2. Convert decimal octal number

3. Convert decimal hexadecimal number

```
# Function to convert a decimal number to binary
binary() {
  read -p "Enter a decimal number: " num
  echo "The binary equivalent of $num is" $(echo "obase=2; $num" | bc)
}
# Function to convert a decimal number to octal
octal() {
  read -p "Enter a decimal number: " num
  echo "The octal equivalent of $num is" $(echo "obase=8; $num" | bc)
}
```

```
# Function to convert a decimal number to hexadecimal
hexadecimal() {
  read -p "Enter a decimal number: " num
  echo "The hexadecimal equivalent of $num is" $(echo
"obase=16; $num" | bc)
}
while :
do
  clear
  echo "1. Convert decimal to binary"
  echo "2. Convert decimal to octal"
  echo "3. Convert decimal to hexadecimal"
  echo "0. Exit"
  read -p "Enter your choice [0-3]: " choice
  # Check the user's input and call the appropriate function
  case $choice in
    1) binary;;
    2) octal;;
    3) hexadecimal;;
    0) exit 0;;
    *) echo "Invalid choice";;
  esac
  read -p "Press [Enter] to continue"
done
```

OUTPUT:-

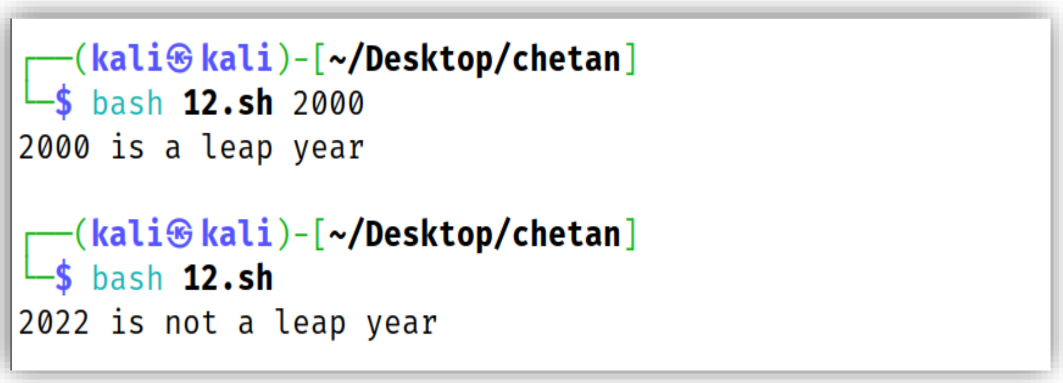


```
1. Convert decimal to binary
2. Convert decimal to octal
3. Convert decimal to hexadecimal
0. Exit
Enter your choice [0-3]: 1
Enter a decimal number: 34
The binary equivalent of 34 is 100010
Press [Enter] to continue
```

12. Write a shell script to check inputted year is leap or not, if no input from user then current year should assumed.

```
year=${1:-$(date +%Y)}
# Check if the year is a leap year
if [[ $year -eq 0 || $(($year % 4)) -ne 0 ]]; then
  echo "$year is not a leap year"
elif [[ $(($year % 100)) -eq 0 && $(($year % 400)) -ne 0 ]];
then
  echo "$year is not a leap year"
else
  echo "$year is a leap year"
fi
```

OUTPUT:-

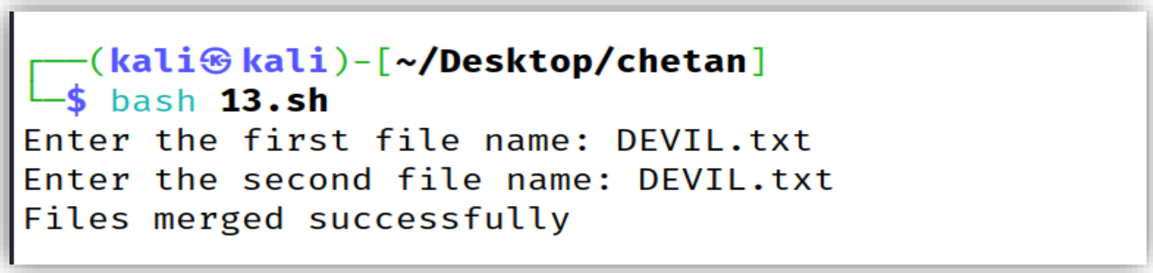


```
┌──(kali㉿kali)-[~/Desktop/chetan]
└─$ bash 12.sh 2000
2000 is a leap year

┌──(kali㉿kali)-[~/Desktop/chetan]
└─$ bash 12.sh
2022 is not a leap year
```

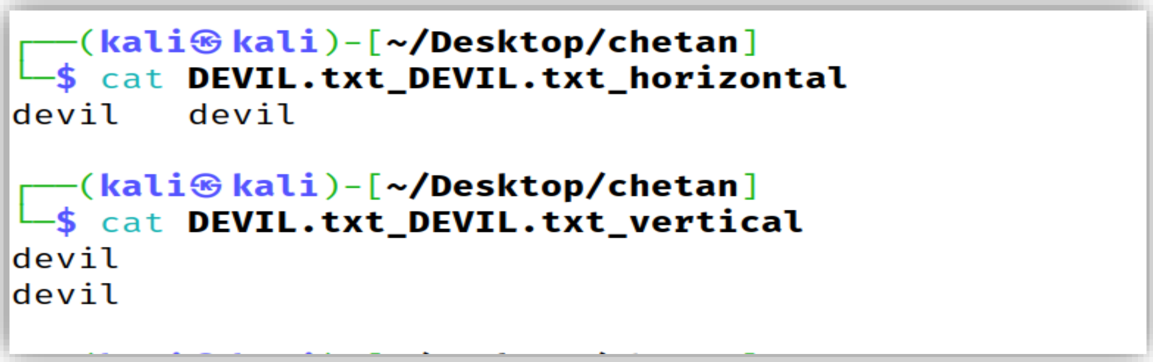
13. Merge 2 files in 1 file horizontally and vertically.

```
# Prompt the user to enter the names of the two files to merge
read -p "Enter the first file name: " file1
read -p "Enter the second file name: " file2
# Use the paste command to merge the two files horizontally
# and write the output to a new file
paste $file1 $file2 > ${file1}_${file2}_horizontal
# Use the cat command to merge the two files vertically
# and write the output to a new file
cat $file1 $file2 > ${file1}_${file2}_vertical
# Display a message indicating that the operation was successful
echo "Files merged successfully"
```

OUTPUT:-



```
┌──(kali㉿kali)-[~/Desktop/chetan]
└─$ bash 13.sh
Enter the first file name: DEVIL.txt
Enter the second file name: DEVIL.txt
Files merged successfully
```



```
┌──(kali㉿kali)-[~/Desktop/chetan]
└─$ cat DEVIL.txt_DEVIL.txt_horizontal
devil    devil

┌──(kali㉿kali)-[~/Desktop/chetan]
└─$ cat DEVIL.txt_DEVIL.txt_vertical
devil
devil
```

14. Write a menu driven script for:

i) Enter 2 strings ii) Display string

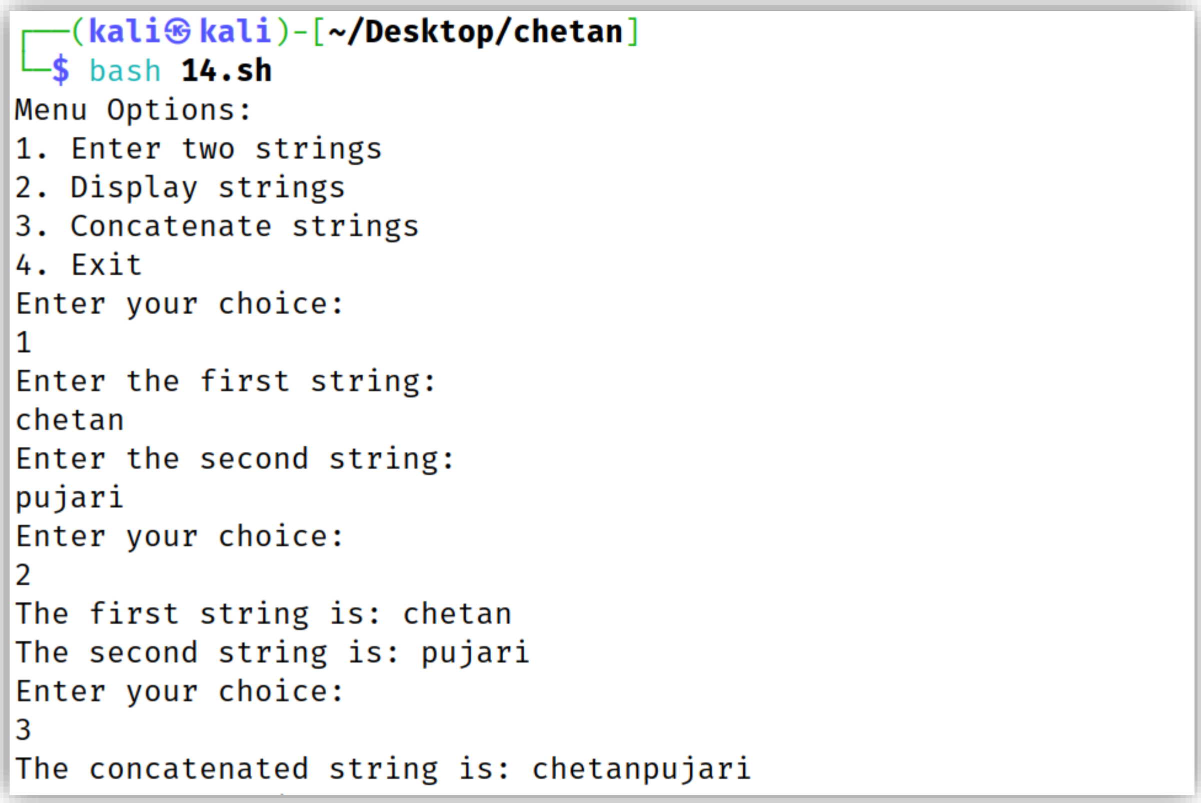
iii) Concatenation of 2 strings iv) Exit

#!/bin/bash

```
# This function is used to display the menu options to the user
display_menu() {
  echo "Menu Options:"
  echo "1. Enter two strings"
  echo "2. Display strings"
  echo "3. Concatenate strings"
  echo "4. Exit"
}

# This function is used to get two strings from the user
get_strings() {
  echo "Enter the first string:"
  read string1

  echo "Enter the second string:"
  read string2
}

# This function is used to display the two entered strings
display_strings() {
  echo "The first string is: $string1"
  echo "The second string is: $string2"
}

# This function is used to concatenate the two entered strings
concatenate_strings() {
  result="$string1$string2"
  echo "The concatenated string is: $result"
}

# Main program

# Display the menu
display_menu

# Initialize the variables
string1=""
string2=""

while true; do
  # Prompt the user to choose an option
  echo "Enter your choice:"
  read choice

  # Perform the selected action
```

```
    case $choice in
      1) get_strings;;
      2) display_strings;;
      3) concatenate_strings;;
      4) exit;;
      *) echo "Invalid choice";;
    esac
  done
```

OUTPUT:-

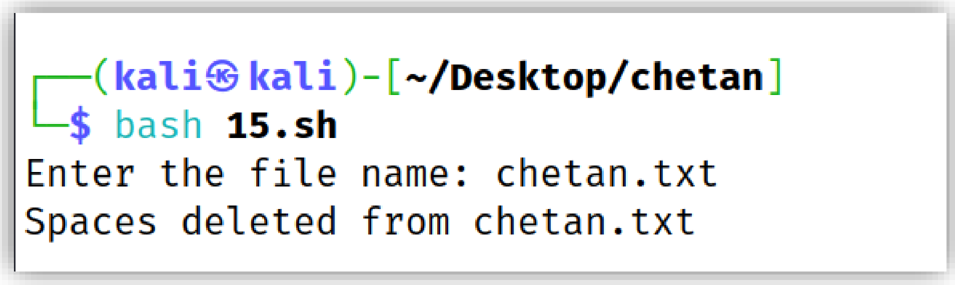


```
┌──(kali㉿kali)-[~/Desktop/chetan]
└─$ bash 14.sh
Menu Options:
1. Enter two strings
2. Display strings
3. Concatenate strings
4. Exit
Enter your choice:
1
Enter the first string:
chetan
Enter the second string:
pujari
Enter your choice:
2
The first string is: chetan
The second string is: pujari
Enter your choice:
3
The concatenated string is: chetanpujari
```

15. Write a shell script to delete all the spaces from given file.

```
#!/bin/bash
# This script deletes all the spaces from a given file.

# Prompt the user to enter the file name
read -p "Enter the file name: " file

# Use the tr command to replace all spaces with nothing (i.e., delete them)
# and write the output to a new file
tr -d ' ' < $file > ${file}_no_spaces

# Display a message indicating that the operation was successful
echo "Spaces deleted from $file"
```

OUTPUT:-

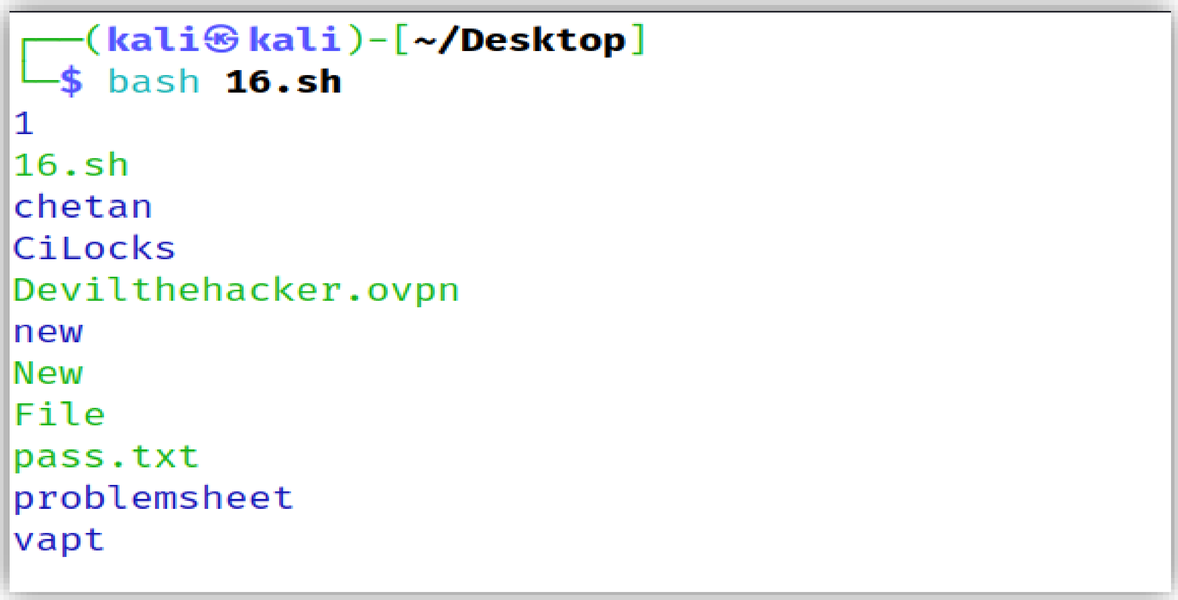


```
┌──(kali㉿kali)-[~/Desktop/chetan]
└─$ bash 15.sh
Enter the file name: chetan.txt
Spaces deleted from chetan.txt
```

16. Write a shell script to display all the files and directory of current directory with proper formatting.

```
files=$(ls)
for file in $files
do
 # Check if the current item is a directory
 if [ -d "$file" ]
 then
  # If it is a directory, display its name in blue
  echo -e "\e[34m$file\e[0m"
 else
  # If it is a file, display its name in green
  echo -e "\e[32m$file\e[0m"
 fi
done
```

OUTPUT:-



```
┌──(kali㉿kali)-[~/Desktop]
└─$ bash 16.sh
1
16.sh
chetan
CiLocks
Devilthehacker.ovpn
new
New
File
pass.txt
problemsheet
vapt
```

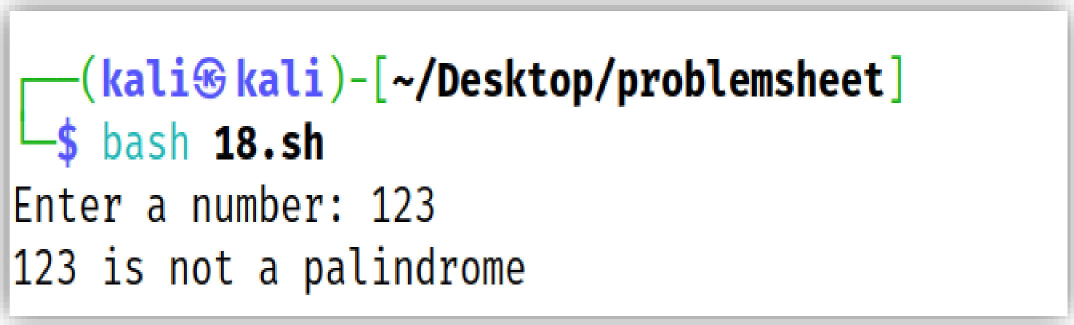
17. Write a shell script program to print multiplication table for given no.

SAME AS QUESTION NO.9

18. Write a shell script to check whether the given number is palindrome or not.

```
# This script checks if the given number is a palindrome.
# Prompt the user to enter a number
read -p "Enter a number: " number

# Store the original number
original=$number

# Initialize the reverse number to 0
reverse=0

# Loop until the number is not 0
while [ $number -ne 0 ]
do
  # Extract the last digit of the number
  last_digit=$(($number % 10))

  # Add the last digit to the reverse number
  reverse=$((reverse * 10 + last_digit))
  # Remove the last digit from the number
  number=$(($number / 10))
done

# If the original number is equal to the reverse number, it is a
palindrome
if [ $original -eq $reverse ]
then
  echo "$original is a palindrome"
else
  echo "$original is not a palindrome"
fi
```

OUTPUT:-



```
┌──(kali㉿kali)-[~/Desktop/problemsheet]
└─$ bash 18.sh
Enter a number: 123
123 is not a palindrome
```

19. Write a menu driven script for:

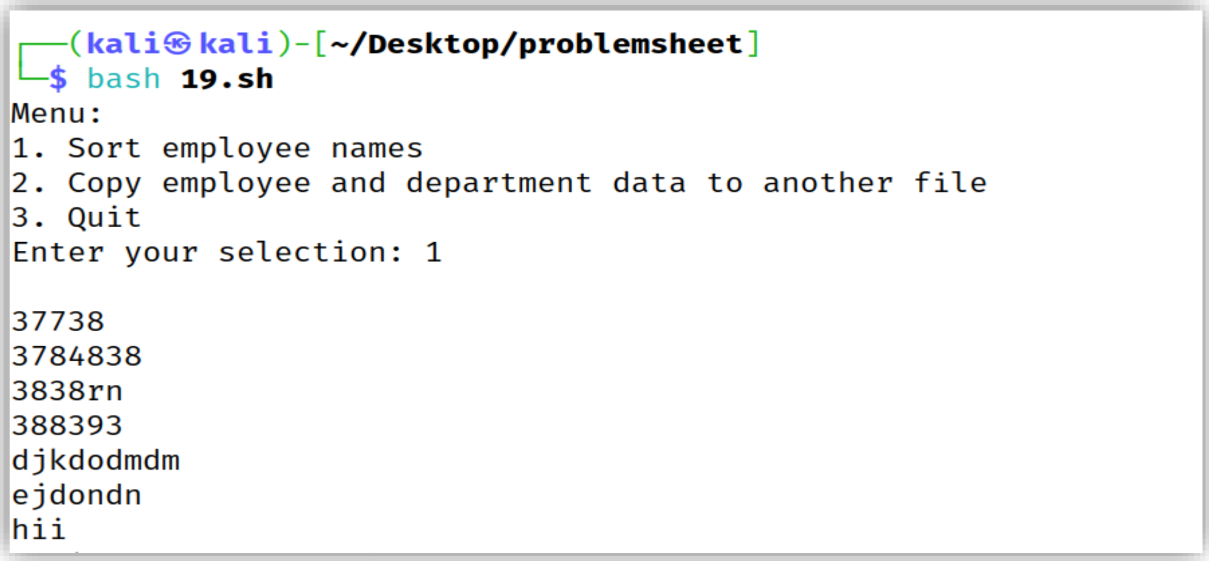
i) Sort employee name in employe.dat file

ii) Copy employee.dat file and department.dat file in another file.

```
# Define the file paths
EMPLOYEE_FILE="./employee.dat"
DEPARTMENT_FILE="./department.dat"
OUTPUT_FILE="./output.dat"
# Display the menu options
echo "Menu:"
echo "1. Sort employee names"
echo "2. Copy employee and department data to another
file"
echo "3. Quit"
# Prompt the user to make a selection
read -p "Enter your selection: " selection
# Process the user's selection
case $selection in
  1)
    # If the user selected 1, sort the employee names
    sort $EMPLOYEE_FILE
    ;;
  2)
 # If the user selected 2, copy the employee and
department data to another file
```

```
        cat $EMPLOYEE_FILE $DEPARTMENT_FILE >
$OUTPUT_FILE
        echo "Data copied to $OUTPUT_FILE"
        ;;
      3)
        # If the user selected 3, exit the script
        exit 0
        ;;
      *)
        # If the user entered an invalid selection, display an
error message
        echo "Invalid selection"
        exit 1
        ;;
    esac
```

OUTPUT:-

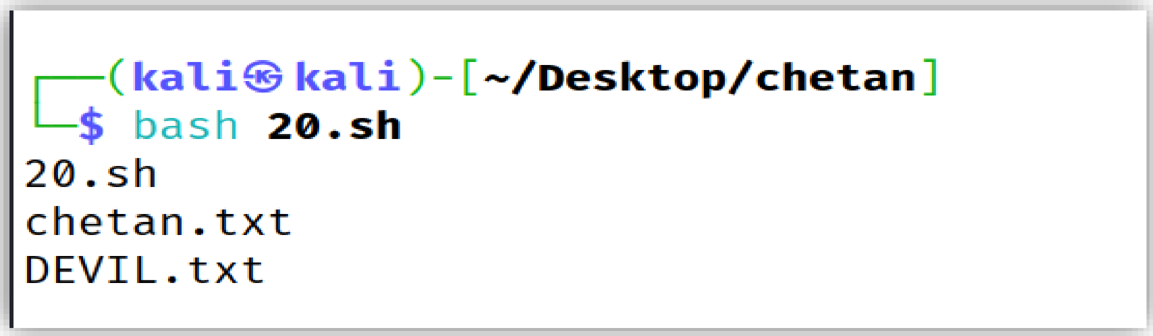


```
┌──(kali㉿kali)-[~/Desktop/problemsheet]
└─$ bash 19.sh
Menu:
1. Sort employee names
2. Copy employee and department data to another file
3. Quit
Enter your selection: 1

37738
3784838
3838rn
388393
djkdodmdm
ejdondn
hii
```

20. Write shell script to display the files of current directory that have the read, write & execute permission for all 3 groups.

```
#!/bin/bash
# This script displays the files in the current directory
# that have read, write, and execute permission for all three groups.

# Loop through all files in the current directory
for file in *
do
  # Check if the file has read, write, and execute permission for all three groups
  if [[ -r "$file" && -w "$file" && -x "$file" ]]
  then
    # If the file has the required permissions, display its name
    echo $file
  fi
done
```

OUTPUT:-



```
┌──(kali㉿kali)-[~/Desktop/chetan]
└─$ bash 20.sh
20.sh
chetan.txt
DEVIL.txt
```