Unix Practical Chapter 7th and 8th 6th May

Add space between , it’s not visible properly in pdf format.

1 :: Display lines containing any digit in the range 1-9, redirect output to output.txt (use grep command)

Answer :: grep -n '[0-9]' hello2 >> hello.txt

-n to print the line number as well .

---

2 :: Display line no. not containing “hello” pattern in it ,redirect output to output.txt(output should be appended to current content of file txt (use grep command)

Answer :: grep -in -v 'hello' hello2 >> hello.txt

If you want to find all lines that do not contain a specified pattern, you can use the -v or --invert-match option.

---

3 :: Display lines of file which can contain at least one occurrence of pattern “hello” in abc.txt and xyz.txt, redirect output to output.txt txt (use grep command)

Answer :: grep -in 'hello' hello2 hello3 >> hello.txt

Note :: -in , -i to ignore case sensitivity , -n to display line number .

The output looks like ::

hello2:4:helloadfadfadf
hello2:7:Adfafddhello
hello2:10:Adfadhello;;fdofdsf
hello2:12:323hello
hello2:17:Hello

hello3:1:How are you hello
hello3:8:What is this hello

hello3:13:Adfad323 helloadfadfa
hello3:17:Adfad3hello

---

4 :: Display date date in mm/dd/yy format redirect output to output.txt

By using :: date +%[format-option]

date "+%D" | cat >> date.txt

OR

date "+%D" >> date.txt

OR

date "+%D" | >> date.txt

---

5 :: Display hours, minute and seconds for current time (use date command)

date "+%T"

---

6 :: Display current month name

date "+%B"

---

7 :: Display content of output.txt (use input redirection)

The output.txt will have some data

cat < output.txt

---

chapter 8th ::

1 ::

Write a function to accept character and return 1 if it vowel and 0 otherwise .
Use this function in main to accept a character and display it Is vowel or not
(use switch case in function to check for vowel)

```
#!/bin/bash

Vowel(){

echo -en "Vowel  Check Script\n"
echo -en "Enter a character to check :: "
read CHAR

case $CHAR in

a)  return 1;;
e)     return 1;;
i)     return 1;;
o)     return 1;;
u)  return 1;;
*)  return 0;;

esac

}
Vowel                # calling the function Vowel
result=$?   # collecting the returned value from function in
                     # variable result

if [ $result -eq 1 ]
then
echo "vowel"
```

```
else
echo "not"
fi
```

2 :: Write a function isEven, which accepts an integer as parameter and returns 1 if the number iseven, and 0 otherwise.
Use this function in main to accept n numbers and check if they are even or odd.

```
#!/bin/bash
even(){

number=$1

if [ $((number%2)) -eq 0 ]
then
return 1
else
return 0
fi

}

even  $1     # calling the function and passing the
             # argument as command line argument
             # so run as :: bash evenCheck
anyNumberYouWantToCheck

RESULT=$?    #collecting the returned value from function

if [ $RESULT -eq 1 ]
then
    echo "even"
else
    echo "odd"
```

```
fi
```

3 ::Write a function isPrime, which accepts an integer as parameter and returns 1 if the number is prime and 0 otherwise.
Use this function in main to display the prime numbers between 1 to 50.

```
#!/bin/bash

prime() {
    current_number=$1
    flag=0
    i=2

    while test $i -le `expr $current_number / 2`
    do
        if test `expr $current_number % $i` -eq 0
        then
            flag=1
        fi
        i=`expr $i + 1`
    done

    if test $flag -eq 0
        then echo $current_number
    fi
}


x=$1
y=$2

for (( number=$x; number<=$y; number++ ))
do
    prime $number
```

```
done
```

4 :: Write a function to accept an integer and return sum of digits of the given number

```
#!/bin/bash
digitSum() {

sum=0
#echo -n "Enter a number "
 num=$1
while [ $num -gt 0 ]
do
    mod=$((num % 10))    #It will split each digits
    sum=$((sum + mod))   #Add each digit to sum
    num=$((num / 10))    #divide num by 10.
done

return $sum
}

digitSum $1

result=$?
echo -e "\nThe sum of digit is :: $result \n"
```

5 :: Write a function to accept two integers x and y and calculate the sum of all integers between x and y (both inclusive)

```
#!/bin/bash

betweenTwoNumbers(){
echo -en "\nEnter x value ::"
read x
```

```
echo -n "Enter y value :: "
read y

n=0
for (( i = $x; i <= $y; i++));
 do
    (( n += i ))
done
return  $n
}

betweenTwoNumbers #Calling the function
result=$?              #Collecting the returned value from function
echo -e "The sum is :: $result\n"
```

6 :: Write a function to accept n display following pattern.

1

1 2

1 2 3

1 2 3 4

And for other related pattern based programs (in shell scripting)

https://www.tutorialsandyou.com/bash-shell-scripting/pyramid-and-pattern-10.html

```
#!/bin/bash

pattern(){
number=1
echo -n "Enter number of rows :: "
read rows
for((i=1; i<=rows; i++))
```

```
do
  for((j=1; j<=i; j++))
  do
    echo -n "$number "
    number=$((number + 1))
  done
  number=1
  echo
done
}
pattern
```