NAME : *VIJAYKUMAR D*

REG.NO: 811519104119

QUESTIONS & ANSWERS..,

(1)Using Any Programming Language of your choice

Write a program in any language of your choice (C, Java, SQL Stored Procedure, Unix Shell Programming, Python). Focus on getting the logic accurate rather than 100% technically correct syntax, as evaluation is more on your ability to logically design the construct.

The problem Statement

The program takes input a short abstract and outputs the following

- prints each word in reverse.
- prints Top 2 occurring words in sorted order. The words that have occurred most of the time.

Sample Input:

"This declaration represents a political commitment among declaration partners to advance a positive vision for the Internet in this era of a united europe"

Sample Output: Printing word in reverse

“Europe of century 21st of era this in Internet the for vision positive a advance to partners declaration among commitment political a represents declaration this”

Sample Output: Occurrence of words

a : 3 times

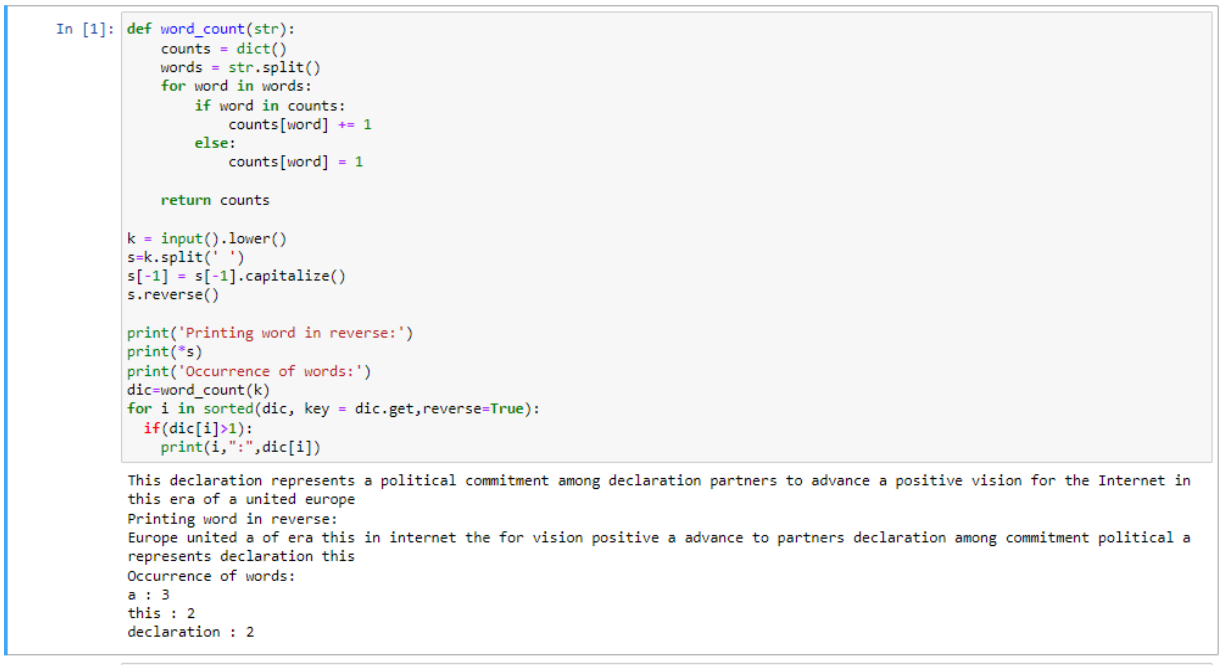
this : 2 times

PROGRAM:

```
def word_count(str):
    counts = dict()
    words = str.split()
    for word in words:
        if word in counts:
            counts[word] += 1
        else:
            counts[word] = 1
    return counts
```

```
k = input().lower()
s=k.split(' ')
s[-1] = s[-1].capitalize()
s.reverse()
print('Printing word in reverse:')
print(*s)
print('Occurrence of words:')
dic=word_count(k)
for i in sorted(dic, key = dic.get,reverse=True):
  if(dic[i]>1):
    print(i,":",dic[i])
```

sample output:



```
In [1]: def word_count(str):
            counts = dict()
            words = str.split()
            for word in words:
                if word in counts:
                    counts[word] += 1
                else:
                    counts[word] = 1

            return counts

        k = input().lower()
        s=k.split(' ')
        s[-1] = s[-1].capitalize()
        s.reverse()

        print('Printing word in reverse:')
        print(*s)
        print('Occurrence of words:')
        dic=word_count(k)
        for i in sorted(dic, key = dic.get,reverse=True):
          if(dic[i]>1):
            print(i,":",dic[i])

        This declaration represents a political commitment among declaration partners to advance a positive vision for the Internet in
        this era of a united europe
        Printing word in reverse:
        Europe united a of era this in internet the for vision positive a advance to partners declaration among commitment political a
        represents declaration this
        Occurrence of words:
        a : 3
        this : 2
        declaration : 2
```

(2) Assignment Using SQL

(i) ASSIGNMENT-01SQL

Please refer to two tables of Appendix A – Purchase History and Product Catalogue. Purchase History contains all purchases done for Grocery Store. Product Catalogue contains all product and its category.

Develop a SQL query that will find out two Products for each product category that are most popular in last 30 days. Popularity is based on maximum quantity sold in a particular category.

Sample Input:

Refer to Table in Appendix A

Sample Output:

Cat_Id	Product_Id	Trending
1	100	1
1	200	2
2	300	1
2	301	2

SQL QUERY:

```
SELECT customer_id,MAX(purch_amt)
FROM orders
GROUP BY customer_id;
```

(ii) ASSIGNMENT-02SQL

There is a Customer Table having a single column with list of customer id. There is a Voucher table having a single column with list of voucher ids.

Develop a query that will assign one voucher to one customer and vice versa. Two customers will not get same voucher. Two Voucher will not be assigned to a single customer.

Sample Input

Customer_Id
Abhinash
Vipin
Mahesh
Bijoy
Bhabani
Ashutosh

Voucher_Id
ABXFH
SDFGH
ERTYY
PPLKM

SQL QUERY:

```
CREATE TABLE Customer (Customer_Id varchar(255));

 INSERT INTO Customer
VALUES('Customer_Id'),('Abhinash'),('Vipin'),('Mahesh'),('Bijoy'),('Bhabani'),('Ashutosh')


 CREATE TABLE Voucher (Voucher_Id varchar(255) UNIQUE);

 INSERT INTO Voucher VALUES('ABXFH'),('SDFGH'),('ERTYY'),('PPLKM')


 ;with cte

 as(select *,row_number() over(order by Customer_Id) rr from Customer)

 ,cte2 as(select *,row_number() over(order by Voucher_Id) rr from Voucher)


 select Customer_Id Customer_Key,Voucher_Id Gift_Voucher_Key

 from cte c1

 join cte2 c2 on c1.rr=c2.rr
```

Assignment Using Unix Shell Programming (Optional)

ASSIGNMENT-01UNIX

There is a data file containing purchase history data. Write a script to find out list of unique dates on which there are sales.

Sample Input :.

Product, Customer, Date, Category, amt
Nylon V Neck, Rakesh, 2020-01-02, Shirt,799.00
Nylon Y Neck, Ramesh, 2020-01-02, Shirt,799.00
Nylon Z Neck, Rajesh, 2020-01-02, Shirt,899.00
Nylon Z Neck, Rajesh, 2020-01-03, Shirt,699.00

Sample Output

2020-01-02
2020-01-03

Unix Scripts:

```
# Script to find unique species in csv files where species is the second data field
# This script accepts any number of file names as command line arguments

# Loop over all files
for file in $@
do
   echo "Unique  $file:"
   # Extract order names
   cut -d , -f 2 $file | sort | uniq
done
```

(ii)ASSIGNMENT-02UNIX

There is a folder that contains everyday purchase history of a Grocery store. There is one file for each day and all files has same format. All files have a header.

Write a unix shell script to merge all files into one file. Merged files must have only 1 header.

Sample Input

/home/grocery/purchase_history_02Jan21.txt
/home/grocery/purchase_history_03Jan21.txt
/home/grocery/purchase_history_31Jan21.txt

Sample Output

/home/grocery/purchase_history_JanAll.txt

UNIX SHELL SCRIPTS:

```
function concat_with_header() {
 # Quoted suffix to pattern match for concatenation (e.g. '*.csv')
 local suffix="${1}"
 # Name of the output file
 local output="${2:-combined.out}"
 # Number of lines to use for the header
 local header_length="${3:-1}"
 # Grab the header from the first file
 local header=`echo -e "$(ls -b *$suffix | head -n$header_length)"`
 head -1 $header_file > $output; tail -n +"`expr $header_length + 1`" -q *$suffix >> $output
}
```

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