Aim= Write a program to demonstrate bitwise operation.

```
a = 60

b = 13

c = 0

c = a & b;

print("Line 1 - Value of c is ", c)

c = a | b;

print ("Line 2 - Value of c is ", c)

c = a ^ b;

print ("Line 3 - Value of c is ", c)

c = ~a;

print ("Line 4 - Value of c is ", c)

c = a << 2;

print ("Line 5 - Value of c is ", c)

c = a >> 2;

print ("Line 6 - Value of c is ", c)
```

Aim = Implement a Page Rank Algorithm.

```
import numpy as np
from scipy.sparse import csc_matrix
from fractions import Fraction

def float_format(vector, decimal):
    return np.round((vector).astype(float), decimals=decimal)

G = np.matrix([[1,1,0],
               [1,0,1],
               [0,1,0]])
n=len(G)
print(n)

M = csc_matrix(G,dtype=float)
rsums = np.array(M.sum(1))[:,0]
ri, ci = M.nonzero()
M.data /= rsums[ri]

dp = Fraction(1,n)

E = np.zeros((3,3))
E[:] = dp

beta = 0.85
```

```
A = beta * M + ((1-beta) * E)


r = np.matrix([dp, dp, dp])
r = np.transpose(r)

previous_r = r
for it in range(1,30):
    r = A * r

    if (previous_r==r).all():
        break
    previous_r = r

print ("Final:\n", float_format(r,3))
print( "sum", np.sum(r))
```

Aim= Implement Dynamic programming algorithm for computing the edit distance between strings s1 and s2. (Hint. Levenshtein Distance)

```
import numpy as np
def levenshtein(s1,s2):
    size_x=len(s1)+1
    size_y=len(s2)+1
    matrix=np.zeros((size_x, size_y))
    for x in range(size_x):
        matrix[x,0]=x
    for y in range(size_y):
        matrix[0,y]=y
    for x in range(1,size_x):
        for y in range(1,size_y):
            if s1[x-1] == s2[y-1]:
                matrix[x,y]=min(matrix[x-1,y]+ 1,matrix[x-1,y-1],matrix[x,y-1]+1)
            else:
                matrix[x,y]=min(matrix[x-1,y]+ 1,matrix[x-1,y-1]+1,matrix[x,y-1]+1)
    print(matrix)
    return(matrix[size_x-1,size_y-1])
levenshtein("Hello","hallo")
```

Aim= Write a program to Compute Similarity between two text documents.

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import numpy as np
from collections import defaultdict
import nltk
def process(file):
    raw = open(file, 'r', encoding='utf-8').read()
    tokens = word_tokenize(raw)
    words = [w.lower() for w in tokens]
    porter = nltk.PorterStemmer()
    Stemmed_tokens = [porter.stem(t) for t in words]
    # Removing stop words
    stop_words = set(stopwords.words('english'))
    filtered_tokens = [w for w in Stemmed_tokens if w not in stop_words]
    # Count words
    count = defaultdict(int)
    for word in filtered_tokens:
        count[word] += 1
    return count

def cos_sim(a, b):
```

```
    dot_product = np.dot(a, b)
    norm_a = np.linalg.norm(a)
    norm_b = np.linalg.norm(b)
    return dot_product / (norm_a * norm_b) if norm_a and norm_b else 0.0


def getSimilarity(dict1, dict2):
    all_words_list = list(set(dict1.keys()).union(set(dict2.keys())))
    v1 = np.zeros(len(all_words_list), dtype=int)
    v2 = np.zeros(len(all_words_list), dtype=int)

    for i, key in enumerate(all_words_list):
        v1[i] = dict1.get(key, 0)
        v2[i] = dict2.get(key, 0)
    return cos_sim(v1, v2)

if __name__ == '__main__':
    nltk.download('punkt')
    nltk.download('stopwords')
    dict1 = process(r"C:\Program Files\Autopsy-4.22.0\README.txt")
    dict2 = process(r"C:\Program Files\Autopsy-4.22.0\NEWS.txt")
    print("Similarity between two text documents:", getSimilarity(dict1, dict2))
```

Aim= Write a program for Pre-processing of a Text Document: stop word removal.

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
example_sent="This is a sample sentence, showing off the stop words filtration."
stop_words=set(stopwords.words('english'))
word_tokens=word_tokenize(example_sent)
filtered_sentence=[w for w in word_tokens if not w in stop_words]
filtered_sentence=[]
for w in word_tokens:
    if w not in stop_words:
        filtered_sentence.append(w)
print(word_tokens)
print(filtered_sentence)
```

Aim= Write a program to implement simple web crawler.

```
import requests
from bs4 import BeautifulSoup

url=("www.amazon.in")
code=requests.get("https://" +url)
plain=code.text
s=BeautifulSoup(plain)
for link in s.find_all("a"):
    print(link.get("href"))
```

Aim= Write a program to parse XML text, generate Web graph and compute topic specific page rank.

```
import csv
import requests
import xml.etree.ElementTree as ET


def loadRSS():
    url = 'http://www.hindustantimes.com/rss/topnews/rssfeed.xml'
    resp = requests.get(url)
    with open('topnewsfeed.xml', 'wb') as f:
        f.write(resp.content)
def parseXML(xmlfile):
    tree = ET.parse(xmlfile)
    root = tree.getroot()
    newsitems=[]
    for item in root.findall('./channel/item'):
        news = {}
        for child in item:
            if child.tag == '{http://search.yahoo.com/mrss/}content':
                news['media']=child.attrib['url']
            else:
                news[child.tag]=child.text.encode('utf8')
                newsitems.append(news)
    return newsitems
def savetoCSV(newsitems, filename):
    fields = ['guid', 'title', 'pubDate', 'description', 'link', 'media']
    with open(filename, 'w') as csvfile:
        writer = csv.DictWriter(csvfile, fieldnames=fields)
        writer.writeheader()
```

```
        writer.writerows(newsitems)
loadRSS()
newsitems = parseXML('topnewsfeed.xml')
savetoCSV(newsitems, 'topnews.csv')
def generate_edges(graph):
    edges=[]
    for node in graph:
        for neighbour in graph[node]:
            edges.append((node,neighbour))
        return edges
```

Aim= Write a program for mining Twitter to identify tweets for a specific period and identify trends and named entities.

Code=

```
import tweepy
consumer_key='rCLpGlj086YIYl3xjz6dwNWTw'
consumer_secret='8dDn10CO6k4HYhg2GIQepYiJXoW8aJ6W2UyvQew2cgupgX4uam'
access_token='1104215432985305089-JzFqwAXhBBdAztqrKTkhFc3RGFLu6r'
access_token_secret='mSdxQ2uLCP0IWUoACCQp1IT8L6sM53RA7N12E5i6y5Oiq'
auth=tweepy.OAuthHandler(consumer_key,consumer_secret)
auth.set_access_token(access_token,access_token_secret)
api=tweepy.API(auth)
public_tweets=api.home_timeline()
for tweet in public_tweets:
    print(tweet.text)
#name="modi"
#tweetCount=10
#results=api.user_timeline(id=name,count=tweetCount)
#for tweet in results:
    #print(tweet.text)
```