NAME :- MOHAN SAI KOTHAPALLI

EMP ID :- 2112951

COHORT CODE :- GN22CDBDS001

ASSIGNMENT ON REGEX

Design a python program to accept a file name through command line arguments.

Parse this file to perform the following:

1. Print all currencies in text, Accepted- $, ₹, £
2. Print all date times in the text- dd/mm/yyyy, dd/mm/yy, mm/dd/yyyy, mm/dd/yy
3. Print all cardinilities and orders- 4th, fifth, sixth, 1st, 2nd, nineteenth, fifth
4. Print all 4 letter words that begin with vowels

STEP 0 :- IMPORTING LIBRARIES

In [1]:
```python
import re
import sys
```

DATA 1

STEP 1 :- Opening File DATA 1

This file contains some random text with currency's and dates and some orders.

In [2]:
```python
f=open('data1.txt')
```

In [3]:
```python
with open('data1.txt','r',encoding='utf-8') as f:
    data=f.readline()
data
```

Out[3]:
```
'Indian currency symbol is ₹. USA currency symbol is $. UK currency symbol is £. 1₹ is 0.013$ and 1$
is 75.95₹ as of 02/04/2022. 1₹ is 0.010£ and 1£ is  99.61£ as of 04/02/2022. 1$ is 76.048₹ as of 28/0
3/22. 1$ is 75.998₹ as of 03/20/22. ISO 8601 allows for date representations without the hyphens sepa
rating the parts of the date. For example 14th June 2021 can be written as 2021-06-14 in the extended
human-readable format but also as 20210614 in the basic format. If agreed upon by both the sender and
receiver, we can modify the year representation to include additional year characters. For example, i
f we'd like to include a 5th year-character and fourteenth hundreeth fifth ours Ours first 1st.\n'
```

The above 'readline()' will read a single line from the file. Line means when you press "enter" in the keyboard will typing your data that is considered as new line. Without pressing you keep on typing then it is considered as single line. i.e '\n'

In [4]:
```python
with open('data1.txt','r',encoding='utf-8') as f:
    data=f.readlines()
data[:5]
```

Out[4]:
```
['Indian currency symbol is ₹. USA currency symbol is $. UK currency symbol is £. 1₹ is 0.013$ and 1$
is 75.95₹ as of 02/04/2022. 1₹ is 0.010£ and 1£ is  99.61£ as of 04/02/2022. 1$ is 76.048₹ as of 28/0
3/22. 1$ is 75.998₹ as of 03/20/22. ISO 8601 allows for date representations without the hyphens sepa
rating the parts of the date. For example 14th June 2021 can be written as 2021-06-14 in the extended
human-readable format but also as 20210614 in the basic format. If agreed upon by both the sender and
receiver, we can modify the year representation to include additional year characters. For example, i
f we'd like to include a 5th year-character and fourteenth hundreeth fifth ours Ours first 1st.\n',
 'In addition, this does not rule out invalid leap days. For example, it will match 02/29/2021 althou
gh 2021 is not a leap year. To do this check will require checking if the year is divisible by 4. Thi
s is not possible with regular expressions and it is recommended that you use the date tools in your
programming language of choice to check for validity.\n',
 'The column Independent Filing Deadline shows the date for the filing of petitions by independent or
third/minor party candidates. This is a general reference\n',
 'date for use by the public and voters. Candidates and others seeking specific information should co
ntact the states for other deadlines that may need\n',
 'to be met. For example, the petitions may have to be checked by officials prior to this date. A dec
laration of candidacy may be due before the petitions are\n']
```

The above 'readlines()' will read all lines from the file as list. Each line as an element in list.

In [5]:
```python
with open('data1.txt','r',encoding='utf-8') as f:
    data=f.read()
data[:1000]
```

Out[5]:
```
'Indian currency symbol is ₹. USA currency symbol is $. UK currency symbol is £. 1₹ is 0.013$ and 1$
is 75.95₹ as of 02/04/2022. 1₹ is 0.010£ and 1£ is  99.61£ as of 04/02/2022. 1$ is 76.048₹ as of 28/0
3/22. 1$ is 75.998₹ as of 03/20/22. ISO 8601 allows for date representations without the hyphens sepa
rating the parts of the date. For example 14th June 2021 can be written as 2021-06-14 in the extended
human-readable format but also as 20210614 in the basic format. If agreed upon by both the sender and
receiver, we can modify the year representation to include additional year characters. For example, i
f we'd like to include a 5th year-character and fourteenth hundreeth fifth ours Ours first 1st.\nIn a
ddition, this does not rule out invalid leap days. For example, it will match 02/29/2021 although 202
1 is not a leap year. To do this check will require checking if the year is divisible by 4. This is n
ot possible with regular expressions and it is recommended that you use the date tools in y'
```

The above 'read()' will read all lines from the file as single para.

STEP 2:- FINDING CURRENCY'S INCLUDING THE AMOUNT IN TEXT

In [6]:
```python
x=re.findall(r"(\d*?\.?\d+ ?[$₹£])",data)
x1=re.findall(r"([$₹£] ?\d*\.?\d+)",data)
print(x+x1)
```

```
['1₹', '0.013$', '1$', '75.95₹', '1₹', '0.010£', '1£', '99.61£', '1$', '76.048₹', '1$', '75.998₹']
```

FINDING ONLY THE SYMBOLS OF CURRENCY IN TEXT

In [7]:
```python
curr=re.findall("([$₹£])",data)
print("Total Number Of Currency Symbols In the TEXT DATA are : ",len(curr))
print(f"Types Of Currency Symbols In the TEXT DATA are : ",len(set(curr))," ",set(curr))
print(curr)
```

```
Total Number Of Currency Symbols In the TEXT DATA are :  15
Types Of Currency Symbols In the TEXT DATA are :  3   {'₹', '$', '£'}
['₹', '$', '£', '₹', '$', '$', '₹', '₹', '£', '£', '£', '$', '₹', '$', '₹']
```

STEP 3 :-PRINTING ALL THE FORMATES OF DATES IN THE TEXT

In [8]:
```python
dates=re.findall(r"((0[1-9]|1[0-2])/(0[1-9]|[12][0-9]|3[01])/(\d{4})\b)",data)
print("The Number of dates in the format of 'mm/dd/yyyy' are : ",len(dates))
for i in range(len(dates)):
    print(dates[i][0],end='  ')
print()
dates=re.findall(r"((0[1-9]|[12][0-9]|3[01])/(0[1-9]|1[0-2])/(\d{4})\b)",data)
print("The Number of dates in the format of 'dd/mm/yyyy' are : ",len(dates))
for i in range(len(dates)):
    print(dates[i][0],end='  ')
print()
dates=re.findall(r"((0[1-9]|[12][0-9]|3[01])/(0[1-9]|1[0-2])/(\d{2})\b)",data)
print("The Number of dates in the format of 'dd/mm/yy' are : ",len(dates))
for i in range(len(dates)):
    print(dates[i][0],end='  ')
print()
dates=re.findall(r"((0[1-9]|1[0-2])/(0[1-9]|[12][0-9]|3[01])/(\d{2})\b)",data)
print("The Number of dates in the format of 'mm/dd/yy' are : ",len(dates))
for i in range(len(dates)):
    print(dates[i][0],end='  ')
print()
```

```
The Number of dates in the format of 'mm/dd/yyyy' are :  12
02/04/2022  04/02/2022  02/29/2021  04/23/2022  03/06/2022  11/08/2022  12/10/2022  07/22/2022  04/2
3/2022  04/23/2022  03/12/2022  03/12/2022
The Number of dates in the format of 'dd/mm/yyyy' are :  7
02/04/2022  04/02/2022  03/06/2022  11/08/2022  12/10/2022  03/12/2022  03/12/2022
The Number of dates in the format of 'dd/mm/yy' are :  3
28/03/22  03/05/20  03/08/22
The Number of dates in the format of 'mm/dd/yy' are :  4
03/20/22  03/05/20  03/19/22  03/08/22
```

STEP 4 :- PRINTING ALL CARDINILITIES AND ORDERS FROM THE TEXT

In [9]:
```python
order=[]
x=re.findall(r"((first|second|third|sixth)|(thir[a-z]+(th|st|nd))|(fou[a-z]+(th|rd|st|nd))|(fif[a-z]+(th|st|nd|rd))|(fi[a-z]+th)|(six[a-z]+(th|st|nd|rd))|(sev[a-z]+(th|st|nd|rd))|(eig[a-z]+(th|st|nd|rd))|(nine[a-z]+(th|st|nd|rd))|(ten[a-z]+?(th|st|nd|rd))|(ele[a-z]+th)|(twe[a-z]+(th|st|nd|rd))|(hun[a-z]+(th|st|rd)))",data)
for i in range(len(x)):
    if(x[i][0] not in order):
        order.append(x[i][0])
#print(order)
orders=[]
x1=re.findall(r"([0-9]+(th|st|nd|rd))",data)
for i in range(len(x1)):
    if(x1[i][0] not in orders):
        orders.append(x1[i][0])
#print(orders)
if order or orders:
    print("The cardinilites and orders present in the data are :\n",order+orders)
else:
    print("No CARDINILITIES AND ORDERS FOUND IN THE TEXT")
```

```
The cardinilites and orders present in the data are :
 ['fourteenth', 'hundreeth', 'fifth', 'first', 'third', '14th', '5th', '1st']
```

STEP 5 :- PRINTING ALL 4 LETTER WORDS STARTING WITH VOWELS FROM THE TEXT

In [10]:
```python
four=re.findall(r"(\b(a|e|i|o|u|A|E|I|O|U)[a-zA-Z]{3}\b)",data)
fours=[]
if four:
    for i in range(len(four)):
        fours.append(four[i][0])
    print(fours)
    print("Tolal number of words without repetition are : ",len(set(fours)),"\n",set(fours))
else:
    print("NO 4 LETTER WORDS STARTING WITH VOWELS FROM THE TEXT")
```

```
['also', 'upon', 'ours', 'Ours', 'Utah', 'Utah']
Tolal number of words without repetition are :  5
 {'also', 'upon', 'ours', 'Utah', 'Ours'}
```

WE WILL REPEAT THE ABOVE STEPS FOR THE REMAINING DATASETS

DATA 2

STEP 1 :- Opening File DATA 2

This TEXT file contains a story and some values of currency's.

In [11]:
```python
f=open('data2.txt')
```

In [12]:
```python
with open('data2.txt','r',encoding='utf-8') as f:
    data=f.readline()
data
```

Out[12]:
```
"'Well, Mr Smith, first if you prefer a different type of nose, we have a large selection availabl
e.'\n"
```

In [13]:
```python
with open('data2.txt','r',encoding='utf-8') as f:
    data=f.readlines()
data[:10]
```

Out[13]:
```
["'Well, Mr Smith, first if you prefer a different type of nose, we have a large selection availabl
e.'\n",
 '\n',
 "'I think this nose is a bit too small.'\n",
 '\n',
 "'Small noses are very fashionable this year, Mr Smith, very fashionable.'\n",
 '\n',
 "'Do you think it suits me?' asked Mr Smith.\n",
 '\n',
 "'I think it second and third looks very nice,' said the shop assistant.\n",
 '\n']
```

In [14]:
```python
with open('data2.txt','r',encoding='utf-8') as f:
    data=f.read()
data[:1000]
```

Out[14]:
```
"'Well, Mr Smith, first if you prefer a different type of nose, we have a large selection availabl
e.'\n\n'I think this nose is a bit too small.'\n\n'Small noses are very fashionable this year, Mr Smi
th, very fashionable.'\n\n'Do you think it suits me?' asked Mr Smith.\n\n'I think it second and third
looks very nice,' said the shop assistant.\n\n'OK, I'll take it!'\n\nOn the airbus home, Mr Smith cal
led his wife on his wristphone.\n\n'Hello dear! Do you like my new nose?'\n\nMrs Smith looked at her
husband's new nose on the videophone monitor on the wall in the kitchen. 'I think it's a bit too smal
l, dear,' she said.\n\n'Small noses are very fashionable this year,' replied Mr Smith, 'very fashiona
ble.' It's all so easy now, thought Mr Smith. A hundred years ago, it was impossible to change your b
ody. Or almost impossible - there was the old-fashioned 'plastic surgery', but it was expensive, pain
ful and dangerous. Ugh! Now, thanks to our 22nd-century genetic engineering, we can change our bodies
when we wa"
```

STEP 2:- FINDING CURRENCY'S INCLUDING THE AMOUNT IN TEXT

In [15]:
```python
x=re.findall(r"(\d*?\.?\d+ ?[$₹£])",data)
x1=re.findall(r"([$₹£] ?\d*\.?\d+)",data)
print(x+x1)
```

```
['100000$', '100000$']
```

FINDING ONLY THE SYMBOLS OF CURRENCY IN TEXT

In [16]:
```python
curr=re.findall("([$₹£])",data)
print("Total Number Of Currency Symbols In the TEXT DATA are : ",len(curr))
print(f"Types Of Currency Symbols In the TEXT DATA are : ",len(set(curr))," ",set(curr))
print(curr)
```

```
Total Number Of Currency Symbols In the TEXT DATA are :  2
Types Of Currency Symbols In the TEXT DATA are :  1   {'$'}
['$', '$']
```

STEP 3 :-PRINTING ALL THE FORMATES OF DATES IN THE TEXT

In [17]:
```python
dates=re.findall(r"((0[1-9]|1[0-2])/(0[1-9]|[12][0-9]|3[01])/(\d{4})\b)",data)
print("The Number of dates in the format of 'mm/dd/yyyy' are : ",len(dates))
for i in range(len(dates)):
    print(dates[i][0],end='  ')
print()
dates=re.findall(r"((0[1-9]|[12][0-9]|3[01])/(0[1-9]|1[0-2])/(\d{4})\b)",data)
print("The Number of dates in the format of 'dd/mm/yyyy' are : ",len(dates))
for i in range(len(dates)):
    print(dates[i][0],end='  ')
print()
dates=re.findall(r"((0[1-9]|[12][0-9]|3[01])/(0[1-9]|1[0-2])/(\d{2})\b)",data)
print("The Number of dates in the format of 'dd/mm/yy' are : ",len(dates))
for i in range(len(dates)):
    print(dates[i][0],end='  ')
print()
dates=re.findall(r"((0[1-9]|1[0-2])/(0[1-9]|[12][0-9]|3[01])/(\d{2})\b)",data)
print("The Number of dates in the format of 'mm/dd/yy' are : ",len(dates))
for i in range(len(dates)):
    print(dates[i][0],end='  ')
print()
```

```
The Number of dates in the format of 'mm/dd/yyyy' are :  0

The Number of dates in the format of 'dd/mm/yyyy' are :  0

The Number of dates in the format of 'dd/mm/yy' are :  0

The Number of dates in the format of 'mm/dd/yy' are :  0
```

STEP 4 :- PRINTING ALL CARDINILITIES AND ORDERS FROM THE TEXT

In [18]:
```python
order=[]
x=re.findall(r"((first|second|third|sixth)|(thir[a-z]+(th|st|nd))|(fou[a-z]+(th|rd|st|nd))|(fif[a-z]+(th|st|nd|rd))|(fi[a-z]+th)|(six[a-z]+(th|st|nd|rd))|(sev[a-z]+(th|st|nd|rd))|(eig[a-z]+(th|st|nd|rd))|(nine[a-z]+(th|st|nd|rd))|(ten[a-z]+?(th|st|nd|rd))|(ele[a-z]+th)|(twe[a-z]+(th|st|nd|rd))|(hun[a-z]+(th|st|rd)))",data)
for i in range(len(x)):
    if(x[i][0] not in order):
        order.append(x[i][0])
#print(order)
orders=[]
x1=re.findall(r"([0-9]+(th|st|nd|rd))",data)
for i in range(len(x1)):
    if(x1[i][0] not in orders):
        orders.append(x1[i][0])
#print(orders)
if order or orders:
    print("The cardinilites and orders present in the data are :\n",order+orders)
else:
    print("No CARDINILITIES AND ORDERS FOUND IN THE TEXT")
```

```
The cardinilites and orders present in the data are :
 ['first', 'second', 'third', '22nd']
```

STEP 5 :- PRINTING ALL 4 LETTER WORDS STARTING WITH VOWELS FROM THE TEXT

In [19]:
```python
four=re.findall(r"(\b(a|e|i|o|u|A|E|I|O|U)[a-zA-Z]{3}\b)",data)
fours=[]
if four:
    for i in range(len(four)):
        fours.append(four[i][0])
    print(fours)
    print("Tolal number of words without repetition are : ",len(set(fours)),"\n",set(fours))
else:
    print("NO 4 LETTER WORDS STARTING WITH VOWELS FROM THE TEXT")
```

```
['easy', 'only', 'else', 'ears', 'Ears', 'ears', 'ears', 'ears', 'eyes', 'arms', 'ears', 'eyes', 'arm
s', 'only', 'used', 'used', 'used', 'eyes', 'into']
Tolal number of words without repetition are :  9
 {'ears', 'only', 'Ears', 'used', 'easy', 'eyes', 'else', 'arms', 'into'}
```

DATA 3

STEP 1 :- Opening File DATA 3

This TEXT file contains INDIAN CONSTUITIONAL AMENDAMENTS with dates and order of the AMENDEMETS.

As there are no currencys in this file i have included some random currency symbols with and without amounts (4)

In [20]:
```python
f=open('data3.txt')
```

In [21]:
```python
with open('data3.txt','r',encoding='utf-8') as f:
    data=f.readline()
data
```

Out[21]:
```
'1st\t15, 19, 85, 87, 174, 176, 341, 342, 372 and 376.Insert articles 31A and 31B.Insert schedule
9.\t18/06/1951 \tAdded special provision for the advancement of any socially and educationally [7] ba
ckward classes or for the Scheduled Castes and Scheduled Tribes (SCs and STs). To fully secure the co
nstitutional validity of zamindari abolition laws and to place reasonable restriction on freedom of s
peech. A new constitutional device, called Schedule 9 introduced to protect against laws that are con
trary to the Constitutionally guaranteed fundamental rights. These laws encroach upon property right
s, freedom of speech and equality before law.\tJawaharlal Nehru\tRajendra Prasad\n'
```

In [22]:
```python
with open('data3.txt','r',encoding='utf-8') as f:
    data=f.readlines()
data[:5]
```

Out[22]:
```
['1st\t15, 19, 85, 87, 174, 176, 341, 342, 372 and 376.Insert articles 31A and 31B.Insert schedule
9.\t18/06/1951 \tAdded special provision for the advancement of any socially and educationally [7] ba
ckward classes or for the Scheduled Castes and Scheduled Tribes (SCs and STs). To fully secure the co
nstitutional validity of zamindari abolition laws and to place reasonable restriction on freedom of s
peech. A new constitutional device, called Schedule 9 introduced to protect against laws that are con
trary to the Constitutionally guaranteed fundamental rights. These laws encroach upon property right
s, freedom of speech and equality before law.\tJawaharlal Nehru\tRajendra Prasad\n',
 '2nd\tAmend article 81(1)(b).[8]\t01/05/1953\tRemoved the upper population limit for a parliamentary
constituency by amending Article 81(1)(b).\n',
 '3rd\tAmend schedule 7.[9]\t22/02/1955\tRe-enacted entry 33 of the Concurrent List in the Seventh Sc
hedule with relation to include trade and commerce in, and the production, supply and distribution of
four classes of essential commodities, viz., foodstuffs, including edible oil seeds and oils; cattle
fodder, including oilcakes and other concentrates; raw cotton whether ginned or unginned, and cotton
seeds; and raw jute. 100$\n',
 '4th\tAmend articles £ 31, 35 and 305.\n',
 'Amend schedule 9.[10]\t27/04/1955\tRestrictions on property rights and inclusion of related bills i
n Schedule 9 of the constitution.\n']
```

In [23]:
```python
with open('data3.txt','r',encoding='utf-8') as f:
    data=f.read()
data[:1000]
```

Out[23]:
```
'1st\t15, 19, 85, 87, 174, 176, 341, 342, 372 and 376.Insert articles 31A and 31B.Insert schedule
9.\t18/06/1951 \tAdded special provision for the advancement of any socially and educationally [7] ba
ckward classes or for the Scheduled Castes and Scheduled Tribes (SCs and STs). To fully secure the co
nstitutional validity of zamindari abolition laws and to place reasonable restriction on freedom of s
peech. A new constitutional device, called Schedule 9 introduced to protect against laws that are con
trary to the Constitutionally guaranteed fundamental rights. These laws encroach upon property right
s, freedom of speech and equality before law.\tJawaharlal Nehru\tRajendra Prasad\n2nd\tAmend article
81(1)(b).[8]\t01/05/1953\tRemoved the upper population limit for a parliamentary constituency by amen
ding Article 81(1)(b).\n3rd\tAmend schedule 7.[9]\t22/02/1955\tRe-enacted entry 33 of the Concurrent
List in the Seventh Schedule with relation to include trade and commerce in, and the production, supp
ly and d'
```

STEP 2:- FINDING CURRENCY'S INCLUDING THE AMOUNT IN TEXT

In [24]:
```python
x=re.findall(r"(\d*?\.?\d+ ?[$₹£])",data)
x1=re.findall(r"([$₹£] ?\d*\.?\d+)",data)
print(x+x1)
```

```
['100$', '100₹', '£ 31']
```

FINDING ONLY THE SYMBOLS OF CURRENCY IN TEXT

In [25]:
```python
curr=re.findall("([$₹£])",data)
print("Total Number Of Currency Symbols In the TEXT DATA are : ",len(curr))
print(f"Types Of Currency Symbols In the TEXT DATA are : ",len(set(curr))," ",set(curr))
print(curr)
```

```
Total Number Of Currency Symbols In the TEXT DATA are :  4
Types Of Currency Symbols In the TEXT DATA are :  3   {'₹', '$', '£'}
['$', '£', '₹', '$']
```

STEP 3 :-PRINTING ALL THE FORMATES OF DATES IN THE TEXT

In [26]:
```python
dates=re.findall(r"((0[1-9]|1[0-2])/(0[1-9]|[12][0-9]|3[01])/(\d{4})\b)",data)
print("The Number of dates in the format of 'mm/dd/yyyy' are : ",len(dates))
for i in range(len(dates)):
    print(dates[i][0],end='  ')
print()
dates=re.findall(r"((0[1-9]|[12][0-9]|3[01])/(0[1-9]|1[0-2])/(\d{4})\b)",data)
print("The Number of dates in the format of 'dd/mm/yyyy' are : ",len(dates))
for i in range(len(dates)):
    print(dates[i][0],end='  ')
print()
dates=re.findall(r"((0[1-9]|[12][0-9]|3[01])/(0[1-9]|1[0-2])/(\d{2})\b)",data)
print("The Number of dates in the format of 'dd/mm/yy' are : ",len(dates))
for i in range(len(dates)):
    print(dates[i][0],end='  ')
print()
dates=re.findall(r"((0[1-9]|1[0-2])/(0[1-9]|[12][0-9]|3[01])/(\d{2})\b)",data)
print("The Number of dates in the format of 'mm/dd/yy' are : ",len(dates))
for i in range(len(dates)):
    print(dates[i][0],end='  ')
print()
```

```
The Number of dates in the format of 'mm/dd/yyyy' are :  26
01/05/1953  12/24/1955  11/09/1956  01/11/1956  05/01/1960  12/28/1960  11/08/1961  01/12/1962  05/1
0/1963  05/10/1963  11/12/1966  10/04/1967  05/11/1971  08/12/1971  09/06/1972  09/06/1972  10/17/197
3  01/06/1974  07/09/1974  01/03/1975  04/26/1975  03/05/1975  01/08/1975  07/08/1976  01/04/1977  0
6/09/1979
The Number of dates in the format of 'dd/mm/yyyy' are :  40
18/06/1951  01/05/1953  22/02/1955  27/04/1955  11/09/1956  01/11/1956  05/01/1960  11/08/1961  19/1
2/1961  20/12/1961  01/12/1962  28/12/1962  05/10/1963  05/10/1963  20/06/1964  27/08/1966  11/12/196
6  22/12/1966  10/04/1967  25/09/1969  23/12/1970  05/11/1971  08/12/1971  28/12/1971  30/12/1971  1
5/02/1972  29/08/1972  09/06/1972  09/06/1972  01/06/1974  19/05/1974  07/09/1974  01/03/1975  03/05/
1975  01/08/1975  07/08/1976  01/04/1977  13/04/1978  06/09/1979  25/01/1980
The Number of dates in the format of 'dd/mm/yy' are :  2
10/08/75  27/05/76
The Number of dates in the format of 'mm/dd/yy' are :  1
10/08/75
```

STEP 4 :- PRINTING ALL CARDINILITIES AND ORDERS FROM THE TEXT

In [27]:
```python
order=[]
x=re.findall(r"((first|second|third|sixth)|(thir[a-z]+(th|st|nd))|(fou[a-z]+(th|rd|st|nd))|(fif[a-z]+(th|st|nd|rd))|(fi[a-z]+th)|(six[a-z]+(th|st|nd|rd))|(sev[a-z]+(th|st|nd|rd))|(eig[a-z]+(th|st|nd|rd))|(nine[a-z]+(th|st|nd|rd))|(ten[a-z]+?(th|st|nd|rd))|(ele[a-z]+th)|(twe[a-z]+(th|st|nd|rd))|(hun[a-z]+(th|st|rd)))",data)
for i in range(len(x)):
    if(x[i][0] not in order):
        order.append(x[i][0])
#print(order)
orders=[]
x1=re.findall(r"([0-9]+(th|st|nd|rd))",data)
for i in range(len(x1)):
    if(x1[i][0] not in orders):
        orders.append(x1[i][0])
#print(orders)
if order or orders:
    print("The cardinilites and orders present in the data are :\n",order+orders)
else:
    print("No CARDINILITIES AND ORDERS FOUND IN THE TEXT")
```

```
The cardinilites and orders present in the data are :
 ['first', '1st', '2nd', '3rd', '4th', '5th', '6th', '7th', '8th', '9th', '10th', '11th', '12th', '13
th', '14th', '15th', '16th', '17th', '18th', '19th', '20th', '21st', '22nd', '23rd', '24th', '25th',
'26th', '27th', '28th', '29th', '30th', '31st', '32nd', '33rd', '34th', '35th', '36th', '37th', '38t
h', '39th', '40th', '41st', '42nd', '43rd', '44th', '45th']
```

STEP 5 :- PRINTING ALL 4 LETTER WORDS STARTING WITH VOWELS FROM THE TEXT

In [28]:
```python
four=re.findall(r"(\b(a|e|i|o|u|A|E|I|O|U)[a-zA-Z]{3}\b)",data)
fours=[]
if four:
    for i in range(len(four)):
        fours.append(four[i][0])
    print(fours)
    print("Tolal number of words without repetition are : ",len(set(fours)),"\n",set(fours))
else:
    print("NO 4 LETTER WORDS STARTING WITH VOWELS FROM THE TEXT")
```

```
['upon', 'oils', 'Also', 'into', 'over', 'away', 'into', 'into', 'acts', 'East', 'acts', 'into', 'act
s', 'Anti']
Tolal number of words without repetition are :  9
 {'away', 'upon', 'acts', 'over', 'East', 'Also', 'Anti', 'into', 'oils'}
```

DATA 4

STEP 1 :- Opening File DATA 4

This TEXT file contains fluctuation of indian currency with dollar and pound daywise.

In [29]:
```python
f=open('data4.txt')
```

In [30]:
```python
with open('data4.txt','r',encoding='utf-8') as f:
    data=f.readline()
data
```

Out[30]:
```
'Date\t                           US Dollar to Indian Rupee\tLink\n'
```

In [31]:
```python
with open('data4.txt','r',encoding='utf-8') as f:
    data=f.readlines()
data[:10]
```

Out[31]:
```
['Date\t                           US Dollar to Indian Rupee\tLink\n',
 'Saturday 2nd April 2022\t1 $ = 75.957 ₹\t$ ₹ rate for 02/04/2022\n',
 'Friday 1st April 2022\t                1 $ = 75.953 ₹\t$ ₹ rate for 01/04/2022\n',
 'Thursday 31st March 2022\t1 $ = 75.910 ₹\t$ ₹ rate for 31/03/2022\n',
 'Wednesday 30th March 2022\t1 $ = 75.825 ₹\t$ ₹ rate for 30/03/2022\n',
 'Tuesday 29th March 2022\t1 $ = 75.667 ₹\t$ ₹ rate for 29/03/2022\n',
 'Monday 28th March 2022\t1 $ = 76.048 ₹\t$ ₹ rate for 28/03/2022\n',
 'Sunday 27th March 2022\t1 $ = 76.287 ₹\t$ ₹ rate for 27/03/2022\n',
 'Saturday 26th March 2022\t1 $ = 76.275 ₹\t$ ₹ rate for 26/03/2022\n',
 'Friday 25th March 2022\t1 $ = 76.276 ₹\t$ ₹ rate for 25/03/2022\n']
```

In [32]:
```python
with open('data4.txt','r',encoding='utf-8') as f:
    data=f.read()
data[:1000]
```

Out[32]:
```
'Date\t                           US Dollar to Indian Rupee\tLink\nSaturday 2nd April 2022\t1 $ = 75.9
57 ₹\t$ ₹ rate for 02/04/2022\nFriday 1st April 2022\t                1 $ = 75.953 ₹\t$ ₹ rate for
01/04/2022\nThursday 31st March 2022\t1 $ = 75.910 ₹\t$ ₹ rate for 31/03/2022\nWednesday 30th March 2
022\t1 $ = 75.825 ₹\t$ ₹ rate for 30/03/2022\nTuesday 29th March 2022\t1 $ = 75.667 ₹\t$ ₹ rate for 2
9/03/2022\nMonday 28th March 2022\t1 $ = 76.048 ₹\t$ ₹ rate for 28/03/2022\nSunday 27th March 2022\t1
$ = 76.287 ₹\t$ ₹ rate for 27/03/2022\nSaturday 26th March 2022\t1 $ = 76.275 ₹\t$ ₹ rate for 26/03/2
022\nFriday 25th March 2022\t1 $ = 76.276 ₹\t$ ₹ rate for 25/03/2022\nThursday 24th March 2022\t1 $ =
76.318 ₹\t$ ₹ rate for 24/03/2022\nWednesday 23rd March 2022\t1 $ = 76.545 ₹\t$ ₹ rate for 23/03/2022
\nTuesday 22nd March 2022\t1 $ = 76.112 ₹\t$ ₹ rate for 03/22/2022\nMonday 21st March 2022\t1 $ = 76.
303 ₹\t$ ₹ rate for 21/03/2022\nSunday 20th March 2022\t1 $ = 75.998 ₹\t$ ₹ rate for 20/03/2022\nSatu
rday 19th March 2022\t1 $ = 75.947 '
```

STEP 2:- FINDING CURRENCY'S INCLUDING THE AMOUNT IN TEXT

In [33]:
```python
x=re.findall(r"(\d*?\.?\d+ ?[$₹£])",data)
x1=re.findall(r"([$₹£] ?\d*\.?\d+)",data)
print(x+x1)
```

```
['1 $', '75.957 ₹', '1 $', '75.953 ₹', '1 $', '75.910 ₹', '1 $', '75.825 ₹', '1 $', '75.667 ₹', '1
$', '76.048 ₹', '1 $', '76.287 ₹', '1 $', '76.275 ₹', '1 $', '76.276 ₹', '1 $', '76.318 ₹', '1 $', '7
6.545 ₹', '1 $', '76.112 ₹', '1 $', '76.303 ₹', '1 $', '75.998 ₹', '1 $', '75.947 ₹', '1 $', '75.947
₹', '1 $', '75.966 ₹', '1 $', '76.020 ₹', '1 $', '76.260 ₹', '1 $', '76.491 ₹', '1 $', '76.753 ₹', '1
$', '76.756 ₹', '1 $', '76.759 ₹', '1 $', '76.293 ₹', '1 $', '76.106 ₹', '1 $', '76.901 ₹', '1 $', '7
7.067 ₹', '1 $', '76.439 ₹', '1 $', '76.416 ₹', '1 $', '76.428 ₹', '1 $', '75.877 ₹', '1 $', '75.593
₹', '1 $', '75.782 ₹', '1 $', '75.289 ₹', '1 $', '75.035 ₹', '1 $', '75.073 ₹', '1 $', '75.053 ₹', '1
$', '75.414 ₹', '1 $', '74.663 ₹', '1 $', '74.643 ₹', '1 $', '74.822 ₹', '1 $', '74.694 ₹', '1 $', '7
4.685 ₹', '1 $', '74.685 ₹', '1 $', '75.090 ₹', '1 $', '74.972 ₹', '1 $', '75.199 ₹', '1 $', '75.687
₹', '1 $', '75.448 ₹', '1 $', '75.640 ₹', '1 $', '75.643 ₹', '1 $', '75.558 ₹', '1 $', '74.811 ₹', '1
$', '74.689 ₹', '1 $', '74.649 ₹', '1 $', '74.650 ₹', '1 $', '74.645 ₹', '1 $', '74.644 ₹', '1 $', '7
4.651 ₹', '1 $', '74.820 ₹', '1 $', '74.763 ₹', '1 $', '74.565 ₹', '1 $', '75.033 ₹', '1 $', '75.002
₹', '1 $', '75.000 ₹', '1 $', '75.214 ₹', '1 $', '75.029 ₹', '1 $', '74.755 ₹', '1 $', '74.640 ₹', '1
$', '74.414 ₹', '1 $', '74.423 ₹', '1 $', '74.423 ₹', '1 $', '74.437 ₹', '1 $', '74.460 ₹', '1 $', '7
4.617 ₹', '1 $', '74.247 ₹', '1 $', '74.270 ₹', '1 $', '74.389 ₹', '1 $', '74.159 ₹', '1 $', '73.972
₹', '1 $', '73.810 ₹', '1 $', '73.821 ₹', '1 $', '74.056 ₹', '1 $', '74.249 ₹', '1 $', '74.232 ₹', '1
$', '74.465 ₹', '1 $', '74.424 ₹', '1 $', '74.430 ₹', '1 $', '74.529 ₹', '1 $', '74.396 ₹', '1 $', '7
4.505 ₹', '1 $', '74.513 ₹', '1 $', '74.511 ₹', '1 $', '74.438 ₹', '1 $', '74.558 ₹', '1 $', '74.721
₹', '1 $', '74.980 ₹', '1 $', '75.397 ₹', '1 $', '75.118 ₹', '1 $', '75.115 ₹', '1 $', '75.066 ₹', '1
$', '75.482 ₹', '1 $', '75.685 ₹', '1 $', '75.744 ₹', '1 $', '76.017 ₹', '1 $', '76.049 ₹', '1 $', '7
6.313 ₹', '1 $', '76.169 ₹', '1 $', '76.230 ₹', '1 $', '76.056 ₹', '1 $', '75.827 ₹', '1 $', '75.739
₹', '1 $', '75.717 ₹', '1 $', '75.792 ₹', '1 $', '75.593 ₹', '1 $', '75.382 ₹', '1 $', '75.407 ₹', '1
$', '75.392 ₹', '1 $', '75.266 ₹', '1 $', '75.240 ₹', '1 $', '75.315 ₹', '1 $', '74.998 ₹', '1 $', '7
5.023 ₹', '1 $', '74.997 ₹', '1 $', '75.068 ₹', '1 $', '75.044 ₹', '1 $', '75.051 ₹', '1 $', '75.056
₹', '1 $', '74.541 ₹', '1 $', '74.622 ₹', '1 $', '74.439 ₹', '1 $', '74.436 ₹', '1 $', '74.312 ₹', '1
$', '74.311 ₹', '1 $', '74.310 ₹', '1 $', '74.186 ₹', '1 $', '74.290 ₹', '1 $', '74.454 ₹', '1 $', '7
4.414 ₹', '1 $', '74.348 ₹', '1 $', '74.357 ₹', '1 $', '74.346 ₹', '1 $', '74.332 ₹', '1 $', '74.405
₹', '1 $', '74.165 ₹', '1 $', '73.918 ₹', '1 $', '74.196 ₹', '1 $', '74.191 ₹', '1 $', '74.191 ₹', '1
$', '74.477 ₹', '1 $', '74.423 ₹', '1 $', '74.698 ₹', '1 $', '74.848 ₹', '1 $', '74.935 ₹', '1 $', '7
4.929 ₹', '1 $', '74.929 ₹', '1 $', '74.778 ₹', '1 $', '75.062 ₹', '1 $', '74.889 ₹', '1 $', '75.096
₹', '1 $', '74.991 ₹', '1 $', '75.002 ₹', '1 $', '74.999 ₹', '1 $', '74.861 ₹', '1 $', '74.831 ₹', '1
$', '75.114 ₹', '1 $', '75.234 ₹', '1 $', '75.023 ₹', '1 $', '74.938 ₹', '1 $', '75.035 ₹', '1 $', '7
5.007 ₹', '1 $', '75.308 ₹', '1 $', '75.490 ₹', '1 $', '75.420 ₹', '1 $', '75.133 ₹', '1 $', '75.134
₹', '1 $', '75.309 ₹', '1 $', '74.854 ₹', '1 $', '74.768 ₹', '1 $', '74.567 ₹', '£1', '₹75.273',
'£1', '₹75.3046', '£1', '₹75.2912', '£1', '₹74.7711', '£1', '₹74.1219', '£1', '₹73.4204', '£1', '₹72.
9117', '£1', '₹72.9769', '£1', '₹73.0007', '£1', '₹73.0041', '£1', '₹73.1103', '£1', '₹73.1562',
'£1', '₹73.7407', '£1', '₹74.2669', '£1', '₹74.4346', '£1', '₹74.4755', '£1', '₹74.4174', '£1', '₹74.
4657', '£1', '₹74.865', '£1', '₹74.8082', '£1', '₹74.6817', '£1', '₹74.7032', '£1', '₹74.579', '£1',
'₹74.6002', '£1', '₹74.5771', '£1', '₹74.302', '£1', '₹74.8179', '£1', '₹74.9475', '£1', '₹74.3339',
'£1', '₹73.9089', '£1', '₹73.9155']
```

FINDING ONLY THE SYMBOLS OF CURRENCY IN TEXT

In [34]:
```python
curr=re.findall("([$₹£])",data)
print("Total Number Of Currency Symbols In the TEXT DATA are : ",len(curr))
print(f"Types Of Currency Symbols In the TEXT DATA are : ",len(set(curr))," ",set(curr))
```

```
Total Number Of Currency Symbols In the TEXT DATA are :  782
Types Of Currency Symbols In the TEXT DATA are :  3   {'₹', '$', '£'}
```

STEP 3 :-PRINTING ALL THE FORMATES OF DATES IN THE TEXT

In [35]:
```python
dates=re.findall(r"((0[1-9]|1[0-2])/(0[1-9]|[12][0-9]|3[01])/(\d{4})\b)",data)
print("The Number of dates in the format of 'mm/dd/yyyy' are : ",len(dates))
for i in range(len(dates)):
    print(dates[i][0],end='  ')
print()
dates=re.findall(r"((0[1-9]|[12][0-9]|3[01])/(0[1-9]|1[0-2])/(\d{4})\b)",data)
print("The Number of dates in the format of 'dd/mm/yyyy' are : ",len(dates))
for i in range(len(dates)):
    print(dates[i][0],end='  ')
print()
dates=re.findall(r"((0[1-9]|[12][0-9]|3[01])/(0[1-9]|1[0-2])/(\d{2})\b)",data)
print("The Number of dates in the format of 'dd/mm/yy' are : ",len(dates))
for i in range(len(dates)):
    print(dates[i][0],end='  ')
print()
dates=re.findall(r"((0[1-9]|1[0-2])/(0[1-9]|[12][0-9]|3[01])/(\d{2})\b)",data)
print("The Number of dates in the format of 'mm/dd/yy' are : ",len(dates))
for i in range(len(dates)):
    print(dates[i][0],end='  ')
print()
```

```
The Number of dates in the format of 'mm/dd/yyyy' are :  84
02/04/2022  01/04/2022  03/22/2022  03/17/2022  12/03/2022  11/03/2022  10/03/2022  09/03/2022  08/0
3/2022  07/03/2022  06/03/2022  05/03/2022  04/03/2022  03/03/2022  02/03/2022  01/03/2022  12/02/202
2  11/02/2022  10/02/2022  09/02/2022  08/02/2022  07/02/2022  06/02/2022  05/02/2022  04/02/2022  0
3/02/2022  02/02/2022  01/02/2022  12/01/2022  11/01/2022  10/01/2022  09/01/2022  08/01/2022  07/01/
2022  06/01/2022  05/01/2022  04/01/2022  03/01/2022  02/01/2022  01/01/2022  12/12/2021  11/12/2021
10/12/2021  09/12/2021  08/12/2021  07/12/2021  06/12/2021  05/12/2021  04/12/2021  03/12/2021  02/1
2/2021  01/12/2021  12/11/2021  11/11/2021  10/11/2021  09/11/2021  08/11/2021  07/11/2021  06/11/202
1  05/11/2021  04/11/2021  03/11/2021  02/11/2021  01/11/2021  12/10/2021  11/10/2021  10/10/2021  0
9/10/2021  08/10/2021  07/10/2021  06/10/2021  05/10/2021  01/01/2010  02/01/2010  03/01/2010  04/01/
2010  05/01/2010  06/01/2010  07/01/2010  08/01/2010  09/01/2010  10/01/2010  11/01/2010  12/01/2010

The Number of dates in the format of 'dd/mm/yyyy' are :  209
02/04/2022  01/04/2022  31/03/2022  30/03/2022  29/03/2022  28/03/2022  27/03/2022  26/03/2022  25/0
3/2022  24/03/2022  23/03/2022  21/03/2022  20/03/2022  19/03/2022  18/03/2022  16/03/2022  15/03/202
2  14/03/2022  13/03/2022  12/03/2022  11/03/2022  10/03/2022  09/03/2022  08/03/2022  07/03/2022  0
6/03/2022  05/03/2022  04/03/2022  03/03/2022  02/03/2022  01/03/2022  28/02/2022  27/02/2022  26/02/
2022  25/02/2022  24/02/2022  23/02/2022  22/02/2022  21/02/2022  20/02/2022  19/02/2022  18/02/2022
17/02/2022  16/02/2022  15/02/2022  14/02/2022  13/02/2022  12/02/2022  11/02/2022  10/02/2022  09/0
2/2022  08/02/2022  07/02/2022  06/02/2022  05/02/2022  04/02/2022  03/02/2022  02/02/2022  01/02/202
2  31/01/2022  30/01/2022  29/01/2022  28/01/2022  27/01/2022  26/01/2022  25/01/2022  24/01/2022  2
3/01/2022  22/01/2022  21/01/2022  20/01/2022  19/01/2022  18/01/2022  17/01/2022  16/01/2022  15/01/
2022  14/01/2022  13/01/2022  12/01/2022  11/01/2022  10/01/2022  09/01/2022  08/01/2022  07/01/2022
06/01/2022  05/01/2022  04/01/2022  03/01/2022  02/01/2022  01/01/2022  31/12/2021  30/12/2021  29/1
2/2021  28/12/2021  27/12/2021  26/12/2021  25/12/2021  24/12/2021  23/12/2021  22/12/2021  21/12/202
1  20/12/2021  19/12/2021  18/12/2021  17/12/2021  16/12/2021  15/12/2021  14/12/2021  13/12/2021  1
2/12/2021  11/12/2021  10/12/2021  09/12/2021  08/12/2021  07/12/2021  06/12/2021  05/12/2021  04/12/
2021  03/12/2021  02/12/2021  01/12/2021  30/11/2021  29/11/2021  28/11/2021  27/11/2021  26/11/2021
25/11/2021  24/11/2021  23/11/2021  22/11/2021  21/11/2021  20/11/2021  19/11/2021  18/11/2021  17/1
1/2021  16/11/2021  15/11/2021  14/11/2021  13/11/2021  12/11/2021  11/11/2021  10/11/2021  09/11/202
1  08/11/2021  07/11/2021  06/11/2021  05/11/2021  04/11/2021  03/11/2021  02/11/2021  01/11/2021  3
1/10/2021  30/10/2021  29/10/2021  28/10/2021  27/10/2021  26/10/2021  25/10/2021  24/10/2021  23/10/
2021  22/10/2021  21/10/2021  20/10/2021  19/10/2021  18/10/2021  17/10/2021  16/10/2021  15/10/2021
14/10/2021  13/10/2021  12/10/2021  11/10/2021  10/10/2021  09/10/2021  08/10/2021  07/10/2021  06/1
0/2021  05/10/2021  01/01/2010  02/01/2010  03/01/2010  04/01/2010  05/01/2010  06/01/2010  07/01/201
0  08/01/2010  09/01/2010  10/01/2010  01/11/2010  12/01/2010  13/01/2010  14/01/2010  15/01/2010  1
6/01/2010  17/01/2010  18/01/2010  19/01/2010  20/01/2010  21/01/2010  22/01/2010  23/01/2010  24/01/
2010  25/01/2010  26/01/2010  27/01/2010  28/01/2010  29/01/2010  30/01/2010  31/01/2010
The Number of dates in the format of 'dd/mm/yy' are :  0

The Number of dates in the format of 'mm/dd/yy' are :  0
```

STEP 4 :- PRINTING ALL CARDINILITIES AND ORDERS FROM THE TEXT

In [36]:
```python
order=[]
x=re.findall(r"((first|second|third|sixth)|(thir[a-z]+(th|st|nd))|(fou[a-z]+(th|rd|st|nd))|(fif[a-z]+(th|st|nd|rd))|(fi[a-z]+th)|(six[a-z]+(th|st|nd|rd))|(sev[a-z]+(th|st|nd|rd))|(eig[a-z]+(th|st|nd|rd))|(nine[a-z]+(th|st|nd|rd))|(ten[a-z]+?(th|st|nd|rd))|(ele[a-z]+th)|(twe[a-z]+(th|st|nd|rd))|(hun[a-z]+(th|st|rd)))",data)
for i in range(len(x)):
    if(x[i][0] not in order):
        order.append(x[i][0])
#print(order)
orders=[]
x1=re.findall(r"([0-9]+(th|st|nd|rd))",data)
for i in range(len(x1)):
    if(x1[i][0] not in orders):
        orders.append(x1[i][0])
#print(orders)
if order or orders:
    print("The cardinilites and orders present in the data are :\n",order+orders)
else:
    print("No CARDINILITIES AND ORDERS FOUND IN THE TEXT")
```

```
The cardinilites and orders present in the data are :
 ['seventh', 'sixth', 'fifth', 'fourth', 'third', 'second', 'first', 'thirtyfirst', 'thirtirth', 'twe
ntyninth', '2nd', '1st', '31st', '30th', '29th', '28th', '27th', '26th', '25th', '24th', '23rd', '22n
d', '21st', '20th', '19th', '18th', '17th', '16th', '15th', '14th', '13th', '12th', '11th', '10th',
'9th', '8th', '7th', '6th', '5th', '4th', '3rd', '23th', '22th', '21th']
```

STEP 5 :- PRINTING ALL 4 LETTER WORDS STARTING WITH VOWELS FROM THE TEXT

In [37]:
```python
four=re.findall(r"(\b(a|e|i|o|u|A|E|I|O|U)[a-zA-Z]{3}\b)",data)
fours=[]
if four:
    for i in range(len(four)):
        fours.append(four[i][0])
    print(fours)
    print("Tolal number of words without repetition are : ",len(set(fours)),"\n",set(fours))
else:
    print("NO 4 LETTER WORDS STARTING WITH VOWELS FROM THE TEXT")
```

```
NO 4 LETTER WORDS STARTING WITH VOWELS FROM THE TEXT
```

DATA 5

STEP 1 :- Opening File DATA 5

This TEXT file contains a book about indian history.

In [38]:
```python
f=open('data5.txt')
```

In [39]:
```python
with open('data5.txt','r',encoding='utf-8') as f:
    data=f.read()
data[:1000]
```

Out[39]:
```
'ANCIENT INDIA\nIndus Valley Civilization\n• Discovered in 1921\n• Belonged to the bronze age\n•  An
area of about 1.3 mn sq km\n• Existed between 3300-1600 BC in three phases: early, mature and late ph
ases\n• Sites\nEarly (pre-Harappan)\nKalibangan Banawali\nDholavira (Kutch) Rakhigarhi (Ghaggar)\nMat
ure (Harappan) Harappa Mohenjodaro Chanhu-daro Lothal\nKalibangan Banawali (Hissar)\nSutkagendor (Pak
istan) Sukotada (Gujarat) Dholavira\nRakhigarhi\nLate phase (post-urban)\nDholavira Rakhigarhi Bhagwa
npura\nManda (Jammu); Chandigarh, Shangol (Punjab); Daulatpur, Mitthal (Haryana); Alamgirpur. Hulas
(West \nUP)\nsite\nSutkagendor — Surkotada Mohenjo-daro\nKalibangan\nRemarkable Feature Marked by a c
itadel Great Bath;\nLarge granary\nImpressive drainage system Piece of woven cotton Mother Goddess\nS
eal of pashu-pati Grain and plough\n• Town planning\no Grid system\n•  The Indus people were the earl
iest to produce cotton\nAryans\nMale dominated Pastoral\nHorse was a significant animal\nRig veda — C
onsists of 10 mandalas '
```

STEP 2:- FINDING CURRENCY'S INCLUDING THE AMOUNT IN TEXT

In [40]:
```python
x=re.findall(r"(\d*?\.?\d+ ?[$₹£])",data)
x1=re.findall(r"([$₹£] ?\d*\.?\d+)",data)
print(x+x1)
```

```
['₹12800', '₹8190']
```

STEP 3 :-PRINTING ALL THE FORMATES OF DATES IN THE TEXT

In [41]:
```python
dates=re.findall(r"((0[1-9]|1[0-2])/(0[1-9]|[12][0-9]|3[01])/(\d{4})\b)",data)
print("The Number of dates in the format of 'mm/dd/yyyy' are : ",len(dates))
for i in range(len(dates)):
    print(dates[i][0],end='  ')
print()
dates=re.findall(r"((0[1-9]|[12][0-9]|3[01])/(0[1-9]|1[0-2])/(\d{4})\b)",data)
print("The Number of dates in the format of 'dd/mm/yyyy' are : ",len(dates))
for i in range(len(dates)):
    print(dates[i][0],end='  ')
print()
dates=re.findall(r"((0[1-9]|[12][0-9]|3[01])/(0[1-9]|1[0-2])/(\d{2})\b)",data)
print("The Number of dates in the format of 'dd/mm/yy' are : ",len(dates))
for i in range(len(dates)):
    print(dates[i][0],end='  ')
print()
dates=re.findall(r"((0[1-9]|1[0-2])/(0[1-9]|[12][0-9]|3[01])/(\d{2})\b)",data)
print("The Number of dates in the format of 'mm/dd/yy' are : ",len(dates))
for i in range(len(dates)):
    print(dates[i][0],end='  ')
print()
```

```
The Number of dates in the format of 'mm/dd/yyyy' are :  10
07/23/1916  01/09/1915  04/13/1919  06/09/1920  01/08/1920  05/02/1922  02/12/1922  02/10/1939  05/0
2/1924  06/11/1924
The Number of dates in the format of 'dd/mm/yyyy' are :  10
01/09/1915  06/09/1920  01/08/1920  17/11/1921  05/02/1922  02/12/1922  02/10/1939  30/03/1924  05/0
2/1924  06/11/1924
The Number of dates in the format of 'dd/mm/yy' are :  0

The Number of dates in the format of 'mm/dd/yy' are :  0
```

STEP 4 :- PRINTING ALL CARDINILITIES AND ORDERS FROM THE TEXT

```
In [42]: order=[]
x=re.findall(r"((first|second|third|sixth)|(thir[a-z]+(th|st|nd))|(fou[a-z]+(th|rd|st|nd))|(fif[a-z]+(th|st|nd|rd))|(fi[a-z]+th)|(six[a-z]+(th|st|nd|rd))|(sev[a-z]+(th|st|nd|rd))|(eig[a-z]+(th|st|nd|rd))|(nine[a-z]+(th|st|nd|rd))|(ten[a-z]+?(th|st|nd|rd))|(ele[a-z]+th)|(twe[a-z]+(th|st|nd|rd))|(hun[a-z]+(th|st|rd)))",data)
for i in range(len(x)):
    if(x[i][0] not in order):
        order.append(x[i][0])
#print(order)
orders=[]
x1=re.findall(r"([0-9]+(th|st|nd|rd))",data)
for i in range(len(x1)):
    if(x1[i][0] not in orders):
        orders.append(x1[i][0])
#print(orders)
if order or orders:
    print("The cardinilites and orders present in the data are :\n",order+orders)
else:
    print("No CARDINILITIES AND ORDERS FOUND IN THE TEXT")
```

```
The cardinilites and orders present in the data are :
 ['first', 'second', 'fifth', 'sixth', 'nineteenth', 'third', 'sixtieth', 'fourth', 'twentieth', 'thi
rteenth', 'eleventh', 'twelfth', 'eighth', 'seventh', '9th', '10th', '12th', '22nd', '31st', '2nd',
'8th', '3rd', '15th', '6th', '4th', '18th', '16th', '19th', '17th', '26th', '1st', '7th', '14th', '11
th', '20th']
```

STEP 5 :- PRINTING ALL 4 LETTER WORDS STARTING WITH VOWELS FROM THE TEXT

```
In [43]: four=re.findall(r"(\b(a|e|i|o|u|A|E|I|O|U)[a-zA-Z]{3}\b)",data)
fours=[]
if four:
    for i in range(len(four)):
        fours.append(four[i][0])
    print(fours)
    print("Tolal number of words without repetition are : ",len(set(fours)),"\n",set(fours))
else:
    print("NO 4 LETTER WORDS STARTING WITH VOWELS FROM THE TEXT")
```

```
['area', 'Indo', 'Iran', 'into', 'Used', 'Used', 'Used', 'Anga', 'also', 'Indo', 'ever', 'Also', 'onl
y', 'Atom', 'Atma', 'into', 'over', 'army', 'East', 'Univ', 'Univ', 'Asia', 'Arab', 'East', 'Asia',
'Asia', 'into', 'used', 'over', 'Amir', 'Each', 'Iran', 'amir', 'also', 'army', 'iqta', 'army', 'use
d', 'Agra', 'Adil', 'Adil', 'Abla', 'Azeb', 'arch', 'Alai', 'Agra', 'used', 'Abul', 'Agra', 'into',
'into', 'into', 'Afif', 'into', 'into', 'also', 'Agra', 'over', 'ease', 'Also', 'army', 'away', 'Ala
m', 'Alam', 'Alam', 'upon', 'East', 'only', 'Asaf', 'Udai', 'over', 'East', 'also', 'army', 'over',
'Iran', 'into', 'into', 'into', 'army', 'anti', 'into', 'army', 'Asia', 'only', 'over', 'aunt', 'ove
r', 'Imad', 'Army', 'Army', 'Asia', 'army', 'army', 'used', 'over', 'also', 'Each', 'over', 'only',
'only', 'only', 'only', 'also', 'used', 'anti', 'anti', 'Only', 'army', 'army', 'into', 'also', 'Oud
h', 'also', 'only', 'only', 'area', 'over', 'over', 'also', 'ones', 'each', 'also', 'into', 'into',
'away', 'only', 'ones', 'only', 'also', 'Urdu', 'East', 'urge', 'used', 'Aims', 'into', 'idea', 'onl
y', 'acts', 'anti', 'also', 'Arya', 'idea', 'upon', 'evil', 'army', 'over', 'idea', 'over', 'Indu',
'anti', 'only', 'into', 'only', 'over', 'army', 'into', 'only', 'also', 'anti', 'over', 'even', 'als
o', 'also', 'used', 'also', 'used', 'also', 'Amar', 'also', 'able', 'over', 'anti', 'Ajit', 'anti',
'even', 'idea', 'Once', 'able', 'also', 'Imam', 'anti', 'Over', 'Urdu', 'into', 'idea', 'Urdu', 'als
o', 'into', 'into', 'each', 'over', 'used', 'Also', 'away', 'also', 'idea', 'open', 'over', 'over',
'also', 'idle', 'only', 'into', 'only', 'aims', 'open', 'used', 'Azad', 'upon', 'away', 'item', 'als
o', 'into', 'Asaf', 'army', 'over', 'anti', 'upon', 'into', 'Acts', 'anti', 'anti', 'also', 'over',
'over', 'used', 'away', 'over', 'othe', 'over', 'ones', 'anti', 'only', 'also', 'Even', 'anti', 'Aza
d', 'also', 'epic', 'into', 'Azad', 'into', 'also', 'over', 'away', 'even', 'also', 'also', 'Over',
'over', 'over', 'also', 'also', 'also', 'over', 'upon', 'even', 'idea', 'over', 'over', 'also', 'ove
r', 'Even', 'into', 'only', 'even', 'used', 'into', 'into', 'iles', 'aced', 'only', 'into', 'also',
'also', 'also', 'into', 'each', 'only', 'also', 'also', 'once', 'only', 'irst', 'ated', 'also', 'int
o', 'also', 'into', 'also', 'even', 'even', 'even', 'upon', 'over', 'also', 'arms', 'also', 'Even',
'only', 'also', 'also', 'upon', 'only', 'over', 'Army', 'Ajoy', 'only', 'only', 'upIi', 'also', 'ide
a', 'also', 'only', 'over', 'Acts', 'Only', 'Even', 'once', 'used', 'over', 'also', 'Idea', 'also',
'only', 'also', 'also', 'idea', 'into', 'akin', 'ices', 'away', 'onIY', 'also', 'acts', 'also', 'als
o', 'also', 'also', 'only', 'upon', 'also', 'idea', 'open', 'also', 'open', 'open', 'ered', 'ound',
'also', 'into', 'even', 'Axis', 'over', 'Asia', 'Army', 'also', 'Azad', 'only', 'also', 'also', 'ide
a', 'also', 'also', 'Army', 'upon', 'Army', 'even', 'also', 'East', 'only', 'even', 'ence', 'army',
'anti', 'also', 'army', 'only', 'Usha', 'Asif', 'Asif', 'also', 'also', 'Ajoy', 'also', 'even', 'use
d', 'only', 'also', 'idea', 'ofto', 'over', 'Udio', 'AZAD', 'also', 'East', 'also', 'Azad', 'also',
'also', 'also', 'anti', 'away', 'army', 'used', 'also', 'also', 'only', 'also', 'army', 'also', 'awa
Y', 'anti', 'East', 'Asia', 'only', 'into', 'only', 'also', 'Only', 'idea', 'also', 'used', 'also',
'used', 'over', 'only', 'ugly', 'also', 'also', 'also', 'into', 'Azad', 'over', 'East', 'also', 'ove
r', 'East', 'acts', 'each', 'Arya', 'over', 'East', 'each', 'East', 'area', 'only', 'area', 'into',
'Only', 'over', 'also', 'Asia', 'also', 'Agni', 'epic', 'into', 'Idrs', 'anti', 'anti', 'into', 'onl
y', 'atma', 'once', 'used', 'only', 'each', 'atom', 'only', 'only', 'over', 'idea', 'into', 'each',
'only', 'only', 'also', 'evil', 'idea', 'else', 'only', 'only', 'also', 'upon', 'also', 'into', 'upo
n', 'Only', 'only', 'into', 'into', 'over', 'into', 'ones', 'idea', 'only', 'Indo', 'into', 'into',
'over', 'into', 'into', 'Isle', 'acts', 'each', 'also', 'even', 'each', 'also', 'into', 'ages', 'int
o', 'atas', 'also', 'also', 'Asia', 'East', 'away', 'each', 'Each', 'into', 'ease', 'East', 'Asia',
'Asia', 'ugly', 'also', 'over', 'open', 'Akal', 'only', 'also', 'Each', 'also', 'Adam', 'over', 'int
o', 'azzz', 'Arab', 'Iraq', 'Iran', 'Imam', 'Omar', 'Imam', 'only', 'Imam', 'Imam', 'Imam', 'Imam',
'used', 'used', 'Arab', 'Adam', 'Aziz', 'Ezra', 'Upon', 'Arab', 'also', 'upon', 'akin', 'Imam', 'Eac
h', 'imam', 'oath', 'into', 'oath', 'alia', 'also', 'over', 'each', 'Arab', 'only', 'only', 'even',
'only', 'Idul', 'Idul', 'Azha', 'over', 'also', 'into', 'into', 'also', 'over', 'East', 'into', 'Eas
t', 'East', 'East', 'East', 'Acts', 'Ares', 'into', 'only', 'over', 'ever', 'into', 'Asia', 'also',
'evil', 'only', 'evil', 'only', 'also', 'into', 'only', 'ages', 'open', 'open', 'each', 'Upon', 'als
o', 'once', 'once', 'each', 'over', 'each', 'even', 'open', 'also', 'idol', 'over', 'Only', 'only',
'also', 'also', 'army', 'Each', 'once', 'only', 'east', 'also', 'also', 'only', 'over', 'only', 'IUC
N', 'also', 'open', 'over', 'IUCN', 'over', 'into', 'Indo', 'Indo', 'Indo', 'Indo', 'Urdu', 'into',
'only', 'over', 'Indo', 'into', 'ayan', 'azzr', 'Abor', 'into', 'over', 'Indo', 'Indo', 'over', 'use
d', 'into', 'used', 'also', 'used', 'over', 'East', 'also', 'only', 'used', 'also', 'over', 'also',
'used', 'even', 'used', 'each', 'used', 'also', 'Only', 'URDU', 'used', 'each', 'used', 'azgc', 'ove
r', 'also', 'also', 'over', 'erac', 'also', 'used', 'area', 'area', 'into', 'used', 'onsg', 'only',
'Each', 'onto', 'used', 'ones', 'into', 'esez', 'into', 'esem', 'also', 'also', 'into', 'used', 'onl
y', 'also', 'East', 'Asia', 'over', 'over', 'ones', 'also', 'upon', 'used', 'idol', 'area', 'also',
'area', 'over', 'atop', 'once', 'Asia', 'each', 'each', 'also', 'over', 'each', 'only', 'used', 'use
d', 'also', 'even', 'also', 'also', 'idal', 'used', 'also', 'into', 'once', 'into', 'also', 'Atop',
'axes', 'also', 'also', 'only', 'away', 'Indo', 'into', 'alas', 'only', 'also', 'open', 'Imam', 'ove
r', 'Indo', 'Indo', 'anew', 'azzz', 'also', 'also', 'arch', 'Indo', 'used', 'also', 'Alai', 'Indo',
'also', 'also', 'amla', 'only', 'also', 'only', 'arch', 'azcb', 'idea', 'used', 'arch', 'into', 'upo
n', 'Indo', 'also', 'also', 'acme', 'Adil', 'arts', 'adZB', 'AZax', 'Agra', 'also', 'also', 'Agra',
'also', 'only', 'also', 'also', 'used', 'Indo', 'ezzr', 'also', 'over', 'used', 'used', 'eyes', 'int
o', 'used', 'also', 'Iran', 'into', 'also', 'each', 'even', 'eyes', 'also', 'even', 'ezed', 'used',
'only', 'Ukil', 'Asit', 'also', 'used', 'into', 'used', 'used', 'into', 'only', 'also', 'axle', 'eac
h', 'only', 'over', 'Each', 'akin', 'into', 'also', 'into', 'each', 'into', 'Adha', 'over', 'only',
'only', 'used', 'unit', 'azxz', 'Alap', 'over', 'Arab', 'Each', 'into', 'into', 'afap', 'over', 'Agr
a', 'also', 'Also', 'also', 'used', 'into', 'Urdu', 'Iran', 'Even', 'Amir', 'upon', 'over', 'over',
'each', 'even', 'each', 'also', 'also', 'into', 'into', 'Urdu', 'Aziz', 'Aziz', 'Aziz', 'also', 'azx
f', 'also', 'epic', 'azra', 'also', 'used', 'also', 'also', 'also', 'idea', 'asas', 'aizz', 'into',
'azrz', 'each', 'into', 'eyes', 'each', 'Each', 'over', 'also', 'into', 'also', 'also', 'able', 'int
o', 'Only', 'away', 'Amma', 'also', 'even', 'into', 'epic', 'used', 'East', 'over', 'open', 'ease',
'into', 'used', 'area', 'edge', 'ilea', 'arts', 'into', 'into', 'used', 'used', 'only', 'also', 'use
d', 'also', 'oral', 'Oxac', 'used', 'into', 'over', 'evil', 'only', 'over', 'into', 'arac', 'Oral',
'used', 'also', 'used', 'also', 'Oval', 'eyes', 'easy', 'iron', 'arms', 'into', 'also', 'used', 'eac
h', 'also', 'also', 'arms', 'into', 'arms', 'into', 'used', 'also', 'each', 'also', 'into', 'over',
'even', 'each', 'also', 'EIIu', 'ends', 'also', 'also', 'also', 'over', 'also', 'azzr', 'also', 'als
o', 'each', 'over', 'also', 'also', 'also', 'Iran', 'Ahom', 'Ahom', 'Ahom', 'open', 'also', 'also',
'also', 'only', 'also', 'epic', 'Asif', 'epic', 'Azam', 'also', 'also', 'also', 'into', 'Oral', 'art
s', 'Oral', 'into', 'each', 'Iran']
Tolal number of words without repetition are :  169
 {'Oval', 'Udio', 'azrz', 'Imad', 'Adha', 'edge', 'Omar', 'ezzr', 'Ahom', 'also', 'unit', 'azxz', 'Ud
ai', 'Over', 'ices', 'Abor', 'Agra', 'Asia', 'azzz', 'Adam', 'Azam', 'ends', 'anew', 'eyes', 'into',
'area', 'Univ', 'east', 'arch', 'Aziz', 'acts', 'Atom', 'azzr', 'urge', 'Ajit', 'evil', 'acme', 'aiz
z', 'idea', 'upon', 'idol', 'aims', 'amir', 'afap', 'alia', 'URDU', 'atop', 'Amir', 'ered', 'Upon',
'Amma', 'item', 'Akal', 'Abla', 'only', 'Isle', 'Adil', 'Ezra', 'Alam', 'Each', 'over', 'ilea', 'els
e', 'Arab', 'arac', 'azxf', 'onIY', 'Axis', 'Used', 'ease', 'ones', 'ofto', 'Acts', 'Alai', 'arts',
'atom', 'Army', 'able', 'Anga', 'iqta', 'open', 'iles', 'easy', 'Azha', 'Iran', 'ated', 'Idrs', 'asa
s', 'Asif', 'Indo', 'esez', 'ezed', 'akin', 'Usha', 'Ukil', 'ages', 'erac', 'Atma', 'Azeb', 'each',
'Asit', 'Only', 'once', 'Amar', 'arms', 'army', 'irst', 'Alap', 'Aims', 'idal', 'ound', 'used', 'ons
g', 'AZax', 'ever', 'East', 'Asaf', 'Iraq', 'idle', 'ugly', 'EIIu', 'Even', 'even', 'axle', 'azra',
'Oxac', 'onto', 'adZB', 'anti', 'Idul', 'iron', 'away', 'Urdu', 'azcb', 'Afif', 'upIi', 'ence', 'AZA
D', 'amla', 'Idea', 'Abul', 'Oudh', 'atma', 'Imam', 'Azad', 'aunt', 'Ares', 'Atop', 'awaY', 'Also',
'oath', 'atas', 'axes', 'othe', 'imam', 'Ajoy', 'Arya', 'alas', 'IUCN', 'Once', 'Agni', 'esem', 'ora
l', 'epic', 'Indu', 'aced', 'Oral', 'azgc', 'ayan'}
```

```
In [ ]:
```