```
"""
Name : Sunil Kumar Puttam
My date of birth is 03/13/2002
as per my month 03 my book is harry potter and prisoner of azkaban

as per date 13 i counted from 14 as there are no page numbers i consider chapther 1 as page 1 and extracted next 10 pages into f
and my year 2002 so from 102 i have extracted next 10 pages into file2.txt

"""
```

Show hidden output

```
from collections import defaultdict
import re

# defining a mapper function
def mapper_word_count(line):
    words = re.findall(r"\b[a-zA-Z']+\b", line.lower())  # extracting words using regrex and all in lower case
    return [(word, 1) for word in words] # returns a list of (word, 1) pairs for each word, in mapreduce the mapper emits (key,

#defining a reducer functiom
def reducer_word_count(mapped_data):
    counts = defaultdict(int) # creating a dictionary with default int = 0
    for word, count in mapped_data:
        counts[word] += count  # adding count for the words
    return counts

# reading the contents of file1
with open("file1.txt", "r", encoding="utf-8") as f:
    lines = f.readlines()

mapped = [] # an empty mapper list
for line in lines:
    mapped.extend(mapper_word_count(line)) #calling the mapper function and loading the mapper list with word pair

word_counts = reducer_word_count(mapped) # passing the mapped list into reducer to get final word counts

print("Word counts in file1.txt:")
for word, count in sorted(word_counts.items(), key=lambda x: (-x[1], x[0])):
    print(f"{word}: {count}") # printing each word with count in a sorted way
```

```
Word counts in file1.txt:
the: 141
to: 85
and: 80
harry: 69
a: 65
his: 57
s: 56
of: 53
he: 51
it: 42
i: 40
was: 40
in: 37
uncle: 36
vernon: 35
on: 34
aunt: 30
you: 29
at: 28
had: 27
for: 24
marge: 24
that: 24
said: 18
she: 17
t: 17
with: 17
be: 16
dudley: 16
ll: 16
up: 16
as: 15
out: 15
petunia: 14
but: 12
```

```
her: 12
him: 12
from: 11
then: 11
your: 11
been: 10
large: 10
book: 9
if: 9
ron: 9
birthday: 8
door: 8
down: 8
form: 8
hagrid: 8
hogwarts: 8
like: 8
next: 8
not: 8
now: 8
off: 8
there: 8
```

```
!pip install pyspellchecker

from spellchecker import SpellChecker
```

```
Collecting pyspellchecker
  Downloading pyspellchecker-0.8.4-py3-none-any.whl.metadata (9.4 kB)
Downloading pyspellchecker-0.8.4-py3-none-any.whl (7.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.2/7.2 MB 32.1 MB/s eta 0:00:00
Installing collected packages: pyspellchecker
Successfully installed pyspellchecker-0.8.4
```

```
from collections import defaultdict
from spellchecker import SpellChecker
import re

spell = SpellChecker()

# creating a mapper functio for non english words
def mapper_non_english(line):
    words = re.findall(r"\b[a-zA-Z']+\b", line)  # storing non english words as pair
    non_english = []  # an empty list
    for word in words:
        if word.lower() not in spell:  # not recognized as English
            non_english.append((word, 1))
    return non_english

# creating a reducer function for non english words
def reducer_non_english(mapped_data):
    counts = defaultdict(int)  # creating a dictionary to count
    for word, count in mapped_data:
        counts[word] += count
    return counts #return dictionary with non-English word count

# reading file2.txt
with open("file2.txt", "r", encoding="utf-8") as f:
    lines = f.readlines()

mapped_ne = []
for line in lines:
    mapped_ne.extend(mapper_non_english(line))

non_english_counts = reducer_non_english(mapped_ne)

print("\nNon-English words in file2.txt:")
for word, count in sorted(non_english_counts.items(), key=lambda x: (-x[1], x[0])):
    print(f"{word}: {count}") # printing each word with count in a sorted way
```

```

Non-English words in file2.txt:
Malfoy: 45
Hagrid: 38
Hermione: 15
Snape: 13
```

```
Buckbeak: 8
wasn: 8
didn: 7
ll: 7
yeh: 7
Goyle: 6
Weasley: 5
shrivelfig: 4
ter: 4
Longbottom: 3
Pomfrey: 3
Slytherins: 3
Tha: 3
hippogriffs: 3
ve: 3
wouldn: 3
Didn: 2
Gryffindor: 2
Seamus: 2
Slytherin: 2
YEH: 2
Yeh: 2
dementors: 2
DOIN: 1
Finnigan: 1
Gryffindors: 1
Hippogriffs: 1
Jus: 1
LETTIN: 1
McGonagall: 1
Muggle: 1
Muggles: 1
Righ: 1
Shoulda: 1
Spect: 1
WANDERIN: 1
alfoy: 1
dyin: 1
flobberworms: 1
hasn: 1
jus: 1
moanin: 1
nors: 1
sayin: 1
walkin: 1
weren: 1
```

```
Start coding or generate with AI.
```