In [1]:
```
#Problem 1

from mrjob.job import MRJob
import re
from collections import defaultdict

WORD_RE = re.compile(r"[\w']+")

def tokenize(text):
    return WORD_RE.findall(text)

def map_reduce(filename):
    word_counts = defaultdict(int)

    with open(filename, 'r') as file:
        for line in file:
            words = tokenize(line)
            for word in words:
                word_counts[word.lower()] += 1

    return word_counts

if __name__ == "__main__":
    filename = "/Users/cheerycheena/Downloads/dtsc701_lab2_prb1.txt"
    word_counts = map_reduce(filename)

    # Print the unique words and their counts
    for word, count in word_counts.items():
        print(f"{word}: {count}")
```

```
lorem: 1
ipsum: 1
dolor: 1
sit: 2
amet: 2
consectetur: 1
adipiscing: 1
elit: 3
donec: 1
condimentum: 1
vel: 1
mauris: 1
varius: 2
id: 1
laoreet: 1
tortor: 1
placerat: 1
nulla: 1
scelerisque: 1
felis: 1
ac: 1
risus: 1
luctus: 1
matti: 1
```

In [2]:
```
#Problem 2

import re

stop_words = ["the", "and", "of", "a", "to", "in", "is", "it"]

def map_non_stop_words(line):
    words = re.sub(r'\W', ' ', line.lower().strip())
    words = words.split()
    non_stop_words = [word for word in words if word not in stop_words]
    word_count = []
    for word in non_stop_words:
        word_count.append((word, 1))
    return word_count

l = "This is a sample input text. It contains some common words such as the,
map_non_stop_words(l)
```

```
Out[2]: [('this', 1),
         ('sample', 1),
         ('input', 1),
         ('text', 1),
         ('contains', 1),
         ('some', 1),
         ('common', 1),
         ('words', 1),
         ('such', 1),
         ('as', 1),
         ('these', 1),
         ('stopwords', 1),
         ('should', 1),
         ('be', 1),
         ('removed', 1),
         ('output', 1)]
```

In [3]:
```
#Problem 3

#importing relevant libraries
import string #to manipulate string variables and remove punctuation marks.

# Define the documents to be inverted
document1 = 'Lorem ipsum dolor sit amet, consectetur adipiscing elit.'
document2 = 'Donec condimentum elit vel mauris varius, id laoreet tortor pla
document3 = 'Nulla scelerisque felis ac risus varius, sit amet luctus elit m

#Remove punctuation marks from the documents
for punctuation in string.punctuation:
    document1 = document1.replace(punctuation, '')
    document2 = document2.replace(punctuation, '')
    document3 = document3.replace(punctuation, '')

# Convert each document to lowercase and split it into words
tokens1 = document1.lower().split()
tokens2 = document2.lower().split()
tokens3 = document3.lower().split()

# Combine the tokens into a list of unique terms
words = list(set(tokens1 + tokens2 + tokens3))

# Create an empty dictionary to store the inverted index
inverted_index = {}

# For each term, find the documents that contain it
for word in words:
    documents = []
    if word in tokens1:
        documents.append("Document 1")
    if word in tokens2:
        documents.append("Document 2")
    if word in tokens3:
        documents.append("Document 3")
    inverted_index[word] = documents

for word, documents in inverted_index.items():
    print(word, "->", ", ".join(documents))
```

```
nulla -> Document 3
risus -> Document 3
consectetur -> Document 1
mauris -> Document 2
scelerisque -> Document 3
varius -> Document 2, Document 3
laoreet -> Document 2
dolor -> Document 1
ipsum -> Document 1
condimentum -> Document 2
vel -> Document 2
placerat -> Document 2
luctus -> Document 3
amet -> Document 1, Document 3
felis -> Document 3
donec -> Document 2
mattis -> Document 3
adipiscing -> Document 1
tortor -> Document 2
sit -> Document 1, Document 3
id -> Document 2
lorem -> Document 1
ac -> Document 3
elit -> Document 1, Document 2, Document 3
```

In [4]:
```
#Problem 4

from collections import Counter
from mrjob.job import MRJob
import re

def map_function(line):
  """Emits each word bigram as a key-value pair, where the key represents th

  words = re.findall(r'\b\w+\b', line.lower())

  for i in range(len(words) - 1):
    bigram = ','.join(words[i:i+2])
    yield bigram, 1

def reduce_function(bigram, counts):
  """Reduces the bigrams by counting their occurrences."""

  counts[bigram] += 1
  return counts

def main():
  """Counts the bigrams in the given text."""

  text = """a man a plan a canal panama there was a plan to build a canal in

  # Split the text into lines.
  lines = text.split('\n')

  # Create a Counter object to store the bigram counts.
  counts = Counter()

  # Map the lines to bigrams.
  for line in lines:
    for bigram, count in map_function(line):
      counts = reduce_function(bigram, counts)

  # Print the bigram counts.
  for bigram, count in counts.items():
    print(f'{bigram}: {count}')

if __name__ == '__main__':
  main()
```

```
a,man: 1
man,a: 1
a,plan: 2
plan,a: 1
a,canal: 3
canal,panama: 1
panama,there: 1
there,was: 1
was,a: 1
plan,to: 1
to,build: 1
build,a: 1
canal,in: 1
in,panama: 2
panama,in: 1
panama,a: 1
canal,was: 1
was,built: 1
```