Experiment No:-6. Write a program that computes the probabilities of n-grams in a corpora. Code this for any value of n. Show results for n=1,2,3. Run this on at least two different publicly available corporas comparing results on the top five most common n-grams in both datasets. Using n-gram probabilities, estimate the probability of an entire sentence. Use log-probabilities to avoid near zero values.

```
import re
import nltk
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.corpus import brown

# Download necessary resources
nltk.download('punkt')
nltk.download('brown')

text = brown.raw()

# Tokenize into sentences
sentences = sent_tokenize(text)

# Convert sentences to lowercase
sentences = [sent.lower() for sent in sentences]

# Process each sentence individually
cleaned_sentences = []
for sent in sentences:
    words = word_tokenize(sent)

    # Remove words containing "/"
    words = [re.sub(r"/.*", "", word) for word in words]

    # Remove all punctuation except full stop (.)
    words = [word for word in words if re.match(r"^\w+$", word) or word == "."]

    # Remove only one punctuation (other than full stops)
    removed_punctuation = False
    new_words = []
    for word in words:
        if not removed_punctuation and re.match(r"[^\w\s]", word) and word != ".":  # Check if it's punctuation (excluding ".")
            removed_punctuation = True  # Remove the first punctuation found
            continue
        new_words.append(word)

    # **REMOVE ALL FULL STOPS (".")**
    new_words = [word for word in new_words if word != "."]

    # Reconstruct the cleaned sentence
    cleaned_sentence = " ".join(new_words).strip()

    if cleaned_sentence:  # Avoid empty sentences
        cleaned_sentences.append(cleaned_sentence)

# Modify sentences by adding "$" and "#"
modified_sentences = []
for sentence in cleaned_sentences:
    modified_sentence = f"$ {sentence} #"
    modified_sentences.append(modified_sentence)

# Print modified sentences
print(modified_sentences[:10])  # Print only the first 10 for preview
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package brown to /root/nltk_data...
[nltk_data]   Package brown is already up-to-date!
['$ the fulton county grand jury said friday an investigation of recent primary election produced no evidence that any irregularitie
```

```
import nltk
from collections import Counter
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.corpus import brown

# Download necessary resources
nltk.download('punkt')
nltk.download('brown')

# Load Brown corpus and preprocess
brown_text = brown.raw()[:5000]  # Limiting for efficiency
sentences = sent_tokenize(brown_text.lower())  # Tokenize into sentences
```

```
# Tokenize the entire corpus into words
tokens = word_tokenize(brown_text.lower())

# Compute Vocabulary Size
V = len(set(tokens))

def compute_unigram_probabilities(tokens):
    """Compute unigram probabilities with Laplace smoothing."""
    unigram_counts = Counter(tokens)
    total_tokens = sum(unigram_counts.values())

    probabilities = {word: (count + 1) / (total_tokens + V) for word, count in unigram_counts.items()}
    return probabilities, unigram_counts

def compute_sentence_probability(sentence, unigram_probs):
    """Compute the probability of a sentence using unigrams."""
    tokens = word_tokenize(sentence.lower())  # Tokenize sentence
    probability = 1.0

    for token in tokens:
        probability *= unigram_probs.get(token, 1 / (sum(unigram_probs.values()) + V))  # Laplace smoothing for unseen words

    return probability

# Compute unigram probabilities from Brown corpus
unigram_probs, _ = compute_unigram_probabilities(tokens)

# Compute probabilities for each sentence
sentence_probs = []
for sentence in cleaned_sentences[:10]:  # Compute for first 10 sentences
    unigram_prob = compute_sentence_probability(sentence, unigram_probs)
    sentence_probs.append((sentence, unigram_prob))

# Display results
for sent, uni_p in sentence_probs:
    print(f"Sentence: {sent}")
    print(f"  Unigram Probability: {uni_p}")
    print("-" * 80)
```

```
Sentence: the fulton county grand jury said friday an investigation of recent primary election produced no evidence that any irregul
  Unigram Probability: 1.9300105927442353e-51
--------------------------------------------------------------------------------
Sentence: the jury further said in presentments that the city executive committee which had charge of the election deserves the prai
  Unigram Probability: 1.1506930050030714e-87
--------------------------------------------------------------------------------
Sentence: the term jury had been charged by fulton superior court judge durwood pye to investigate reports of possible irregularitie
  Unigram Probability: 2.4029584246292775e-68
--------------------------------------------------------------------------------
Sentence: only a relative handful of such reports was received the jury said considering the widespread interest in the election the
  Unigram Probability: 9.242147787035683e-71
--------------------------------------------------------------------------------
Sentence: the jury said it did find that many of registration and election laws are outmoded or inadequate and often ambiguous
  Unigram Probability: 5.018027541135012e-49
--------------------------------------------------------------------------------
Sentence: it recommended that fulton legislators act to have these laws studied and revised to the end of modernizing and improving
  Unigram Probability: 1.9300105927442353e-51
--------------------------------------------------------------------------------
Sentence: the grand jury commented on a number of other topics among them the atlanta and fulton county purchasing departments which
  Unigram Probability: 6.546956104933271e-95
--------------------------------------------------------------------------------
Sentence: merger proposed however the jury said it believes these two offices should be combined to achieve greater efficiency and r
  Unigram Probability: 1.0980943290533885e-58
--------------------------------------------------------------------------------
Sentence: the city purchasing department the jury said is lacking in experienced clerical personnel as a result of city personnel po
  Unigram Probability: 5.018027541135012e-49
--------------------------------------------------------------------------------
Sentence: it urged that the city take steps to remedy this problem
  Unigram Probability: 2.724539899579543e-27
--------------------------------------------------------------------------------
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package brown to /root/nltk_data...
[nltk_data]   Package brown is already up-to-date!
```

```
import nltk
import math
from collections import Counter
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.corpus import brown

# Download necessary resources
```

```
nltk.download('punkt')
nltk.download('brown')

# Load Brown corpus and preprocess
brown_text = brown.raw()[:5000]  # Limiting for efficiency
sentences = sent_tokenize(brown_text.lower())  # Tokenize into sentences

# Tokenize the entire corpus into words
tokens = word_tokenize(brown_text.lower())

# Compute Vocabulary Size
V = len(set(tokens))

def compute_bigram_log_probabilities(tokens):
    """Compute log bigram probabilities with Laplace smoothing."""
    unigram_counts = Counter(tokens)
    bigram_counts = Counter(zip(tokens[:-1], tokens[1:]))

    probabilities = {
        (w1, w2): math.log((count + 1) / (unigram_counts[w1] + V))  # Log applied
        for (w1, w2), count in bigram_counts.items()
    }

    # Define a default log probability for unseen bigrams
    unseen_log_prob = math.log(1 / (sum(unigram_counts.values()) + V))

    return probabilities, unseen_log_prob

def preprocess_sentence(sentence):
    """Prepares sentence by adding start and end tokens."""
    tokens = ["$"] + word_tokenize(sentence.lower()) + ["#"]
    return tokens

def compute_sentence_log_probability(sentence, bigram_log_probs, unseen_log_prob):
    """Compute the log probability of a sentence using bigrams."""
    tokens = preprocess_sentence(sentence)
    total_log_prob = 0.0

    for i in range(len(tokens) - 1):
        bigram = (tokens[i], tokens[i + 1])
        log_prob = bigram_log_probs.get(bigram, unseen_log_prob)  # Use precomputed unseen probability
        total_log_prob += log_prob

    return total_log_prob

# Compute bigram log probabilities from Brown corpus
bigram_log_probs, unseen_log_prob = compute_bigram_log_probabilities(tokens)

# Compute log probabilities for each sentence
sentence_probs = []
for sentence in cleaned_sentences[:10]:  # Compute for first 10 sentences
    bigram_log_prob = compute_sentence_log_probability(sentence, bigram_log_probs, unseen_log_prob)
    sentence_probs.append((sentence, bigram_log_prob))

# Display results
for sent, log_prob in sentence_probs:
    exp_prob = math.exp(log_prob)  # Convert back to probability
    print(f"Sentence: {sent}")
    print(f"  Log Bigram Probability: {log_prob}")  # This will be negative
    print(f"  Exponentiated Bigram Probability: {exp_prob}")  # This will be between 0 and 1
    print("-" * 80)
```

```
Sentence: the fulton county grand jury said friday an investigation of recent primary election produced no evidence that any irregul
  Log Bigram Probability: -150.2316655761171
  Exponentiated Bigram Probability: 5.691367547284337e-66
--------------------------------------------------------------------------------
Sentence: the jury further said in presentments that the city executive committee which had charge of the election deserves the prai
  Log Bigram Probability: -252.66234665074245
  Exponentiated Bigram Probability: 1.8626749106165502e-110
--------------------------------------------------------------------------------
Sentence: the term jury had been charged by fulton superior court judge durwood pye to investigate reports of possible irregularitie
  Log Bigram Probability: -198.03265007760893
  Exponentiated Bigram Probability: 9.897211124644977e-87
--------------------------------------------------------------------------------
Sentence: only a relative handful of such reports was received the jury said considering the widespread interest in the election the
  Log Bigram Probability: -204.86136214925062
  Exponentiated Bigram Probability: 1.071126750914422e-89
--------------------------------------------------------------------------------
Sentence: the jury said it did find that many of registration and election laws are outmoded or inadequate and often ambiguous
  Log Bigram Probability: -143.40295350447542
  Exponentiated Bigram Probability: 5.258823613690755e-63
--------------------------------------------------------------------------------
Sentence: it recommended that fulton legislators act to have these laws studied and revised to the end of modernizing and improving
```

```
  Log Bigram Probability: -150.2316655761171
  Exponentiated Bigram Probability: 5.691367547284337e-66
--------------------------------------------------------------------------------
Sentence: the grand jury commented on a number of other topics among them the atlanta and fulton county purchasing departments which
  Log Bigram Probability: -273.1484828656675
  Exponentiated Bigram Probability: 2.361136806254413e-119
--------------------------------------------------------------------------------
Sentence: merger proposed however the jury said it believes these two offices should be combined to achieve greater efficiency and r
  Log Bigram Probability: -170.71780179104218
  Exponentiated Bigram Probability: 7.214408331385591e-75
--------------------------------------------------------------------------------
Sentence: the city purchasing department the jury said is lacking in experienced clerical personnel as a result of city personnel pc
  Log Bigram Probability: -143.40295350447542
  Exponentiated Bigram Probability: 5.258823613690755e-63
--------------------------------------------------------------------------------
Sentence: it urged that the city take steps to remedy this problem
  Log Bigram Probability: -81.94454485970022
  Exponentiated Bigram Probability: 2.5818817359051712e-36
--------------------------------------------------------------------------------
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package brown to /root/nltk_data...
[nltk_data]   Package brown is already up-to-date!
```

```
import nltk
import math
from collections import Counter
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.corpus import brown

# Download necessary resources
nltk.download('punkt')
nltk.download('brown')

# Load Brown corpus and preprocess
brown_text = brown.raw()[:5000]  # Limiting for efficiency
sentences = sent_tokenize(brown_text.lower())  # Tokenize into sentences

# Tokenize the entire corpus into words
tokens = word_tokenize(brown_text.lower())

# Compute Vocabulary Size
V = len(set(tokens))

def compute_trigram_log_probabilities(tokens):
    """Compute trigram log probabilities with Laplace smoothing."""
    unigram_counts = Counter(tokens)
    bigram_counts = Counter(zip(tokens[:-1], tokens[1:]))
    trigram_counts = Counter(zip(tokens[:-2], tokens[1:-1], tokens[2:]))

    # Compute log P(w3 | w1, w2) = log((Count(w1, w2, w3) + 1) / (Count(w1, w2) + V))
    log_probabilities = {
        (w1, w2, w3): math.log((count) / (bigram_counts[(w1, w2)]))
        for (w1, w2, w3), count in trigram_counts.items()
    }

    return log_probabilities, trigram_counts

def preprocess_sentence(sentence):
    """Add two start symbols ($$) and two end symbols (##) to a sentence."""
    tokens = ["$", "$"] + word_tokenize(sentence.lower()) + ["#", "#"]
    return tokens

def compute_sentence_log_probability(sentence, trigram_log_probs):
    """Compute the log probability of a sentence using trigrams."""
    tokens = preprocess_sentence(sentence)
    total_log_prob = 0.0  # Start at 0 since we sum logs

    for i in range(len(tokens) - 2):
        trigram = (tokens[i], tokens[i + 1], tokens[i + 2])

        smoothing_denominator = sum(trigram_log_probs.values()) + V
        if smoothing_denominator > 0:
            log_prob = trigram_log_probs.get(trigram, math.log(1 / smoothing_denominator))
        else:
            log_prob = math.log(1 / V)  # Avoid log(0) error

        total_log_prob += log_prob  # Summing log probabilities

    return total_log_prob

# Compute trigram log probabilities from Brown corpus
trigram_log_probs, _ = compute_trigram_log_probabilities(tokens)
```

```
# Compute log probabilities for each sentence
sentence_probs = []
for sentence in cleaned_sentences[:10]:  # Compute for first 10 sentences
    trigram_log_prob = compute_sentence_log_probability(sentence, trigram_log_probs)
    sentence_probs.append((sentence, trigram_log_prob))

# Display results
for sent, log_prob in sentence_probs:
    adjusted_log_prob = -log_prob  # Convert to non-negative log probability
    exp_prob = math.exp(log_prob)  # Convert back to probability

    print(f"Sentence: {'$ $ ' + sent +' # #'}")
    print(f"  Adjusted Log Probability: {adjusted_log_prob}")  # This will be positive
    print(f"  Probability: {exp_prob}")  # This will be between 0 and 1
    print("-" * 80)
```

```
Sentence: $ $ the fulton county grand jury said friday an investigation of recent primary election produced no evidence that any irr
  Adjusted Log Probability: 103.58257450439618
  Probability: 1.0343308339088894e-45
--------------------------------------------------------------------------------
Sentence: $ $ the jury further said in presentments that the city executive committee which had charge of the election deserves the
  Adjusted Log Probability: 171.1364274420459
  Probability: 4.746723224447322e-75
--------------------------------------------------------------------------------
Sentence: $ $ the term jury had been charged by fulton superior court judge durwood pye to investigate reports of possible irregular
  Adjusted Log Probability: 135.1077058752994
  Probability: 2.1060509608411885e-59
--------------------------------------------------------------------------------
Sentence: $ $ only a relative handful of such reports was received the jury said considering the widespread interest in the electior
  Adjusted Log Probability: 139.6112960711427
  Probability: 2.3312266808189253e-61
--------------------------------------------------------------------------------
Sentence: $ $ the jury said it did find that many of registration and election laws are outmoded or inadequate and often ambiguous #
  Adjusted Log Probability: 99.07898430855286
  Probability: 9.344236939739686e-44
--------------------------------------------------------------------------------
Sentence: $ $ it recommended that fulton legislators act to have these laws studied and revised to the end of modernizing and improv
  Adjusted Log Probability: 103.58257450439618
  Probability: 1.0343308339088894e-45
--------------------------------------------------------------------------------
Sentence: $ $ the grand jury commented on a number of other topics among them the atlanta and fulton county purchasing departments v
  Adjusted Log Probability: 184.64719802957586
  Probability: 6.437849159846999e-81
--------------------------------------------------------------------------------
Sentence: $ $ merger proposed however the jury said it believes these two offices should be combined to achieve greater efficiency a
  Adjusted Log Probability: 117.09334509192612
  Probability: 1.4028342448509906e-51
--------------------------------------------------------------------------------
Sentence: $ $ the city purchasing department the jury said is lacking in experienced clerical personnel as a result of city personne
  Adjusted Log Probability: 99.07898430855286
  Probability: 9.344236939739686e-44
--------------------------------------------------------------------------------
Sentence: $ $ it urged that the city take steps to remedy this problem # #
  Adjusted Log Probability: 58.546672545963034
  Probability: 3.745442890835569e-26
--------------------------------------------------------------------------------
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package brown to /root/nltk_data...
[nltk_data]   Package brown is already up-to-date!
```