Information Retrieval
Assignment-1

Akash Rawat (MT21005)
Shubham Rana (MT21092)

Question 1:

For this question, we have used the following python libraries:

- Pandas, Numpy.
- NLTK library for pre-processing the data.

Pre-Processing:

1. Read data files from the directory using os.listdir() function.
2. Opened all the files and read the content inside them and stored them in a dataframe.
3. Converted all the words in lower case.
4. Removed all the special characters and punctuation marks.
5. Removed all the stopwords from the corpus using NLTK stopwords.
6. Performed tokenization on the document corpus.
7. Performed lemmatization on the tokens of the corpus.

Methodology:

1. First, we have made a list of all the unique tokens in the corpus.
2. After that, we have initialized our dictionary with all the unique words in the corpus. This dictionary will be used to make the inverted index posting list for the terms later.
3. Now we have scanned all the documents in the increasing order of the document id and while scanning we have made the posting list for all the terms.
4. After the posting list has been created, we have now handled the input query.
5. For handling the input query, first, we have pre-processed the input query as stated in the pre-processing section above.
6. Then we have taken the list of operators that are to be performed in the input query.
7. Now using the input query and the list of operators, we have performed the operations and then fetched the desired result.

List of operations:

- **AND operation**
    - The AND operator function takes 2 posting lists as an argument.
    - These posting lists are already sorted in ascending order of doc ids.

    - We have used 2 pointer approach to intersect both the posting lists. If we found the equal doc ids, we append it in the result, otherwise, we increment the pointer of smaller doc id posting list.
    - While we are comparing, we have also maintained one variable for the comparison where we increment the variable as we go on comparing the posting list.

- **OR operation:**
    - The OR operator function takes 2 posting lists as an argument.
    - These posting lists are already sorted in ascending order of doc ids.
    - We have used 2 pointer approach to find the union of both the posting lists. If we found the equal doc ids, we append it in the result, otherwise, we append and increment the pointer of smaller doc id posting list.
    - While we are comparing, we have also maintained one variable for the comparison where we increment the variable as we go on comparing the posting list.

- **NOT operation:**
    - The NOT operator function takes 1 posting list as an argument.
    - The posting list is already sorted in ascending order of doc ids.
    - We have found the difference between the list that contains all doc ids and this posting list and then return the result.

- **AND NOT operation:**
    - The AND NOT operator function takes 2 posting lists A and B as an argument.
    - These posting lists are already sorted in ascending order of doc ids.
    - First, we calculate the NOT of posting list B from the NOT operation function and store the immediate result.
    - After that, we perform the AND operation between posting list A and the immediate NOT result of posting list B.

- **OR NOT operation:**
    - The OR NOT operator function takes 2 posting lists A and B as an argument.
    - These posting lists are already sorted in ascending order of doc ids.
    - First, we calculate the NOT of posting list B from the NOT operation function and store the immediate result.
    - After that, we perform the OR operation between posting list A and the immediate NOT result of posting list B.

Question 2:

For this question, we have used the following python libraries:

- Pandas, Numpy.
- NLTK library for pre-processing the data.

Pre-Processing:

1. Read data files from the directory using os.listdir() function.
2. Opened all the files and read the content inside them and stored them in a dataframe.
3. Converted all the words in lower case.
4. Removed all the special characters and punctuation marks.
5. Removed all the stopwords from the corpus using NLTK stopwords.
6. Performed tokenization on the document corpus.

Methodology:

1. First, we have made a list of all the unique tokens in the corpus.
2. After that, we have initialized our dictionary with all the unique words in the corpus. This dictionary will be used to make the positional index list for the terms later.
3. Now we have scanned all the documents in the increasing order of the document id and while scanning we have made the posting list for all the terms along with the position of that term in that document.
4. After the positional index list has been created, we have now handled the input query.
5. For handling the input query, first, we have pre-processed the input query as stated in the pre-processing section above.
6. After pre-processing the query we have checked the total number of tokens retrieved. If the number of tokens is greater than 5 then we simply ignore the query, otherwise, we process the query according to the number of tokens.

Conditions:

- **The number of tokens = 0:** No document retrieved.
- **The number of tokens = 1:** Simply return the documents containing the token.
- **The number of tokens = 2:** We check the following condition on all the documents:
  Position of token2 - Position of token 1 == 1
- **The number of tokens = 3:** We check the following condition on all the documents:

Position of token3 - Position of token 2 == 1 **and** Position of token2 - Position of token 1 == 1 **and** Position of token 3 - Position of token 1 ==2.

- **The number of tokens = 4:** We check the following condition on all the documents:
  Position of token 4 - Position of token 3 == 1 **and** Position of token 3 - Position of token 2 == 1 **and** Position of token 2 - Position of token 1 ==1 **and** position of token 4 - Position of token 1 == 3.
- **The number of tokens = 5:** We check the following condition on all the documents:
  Position of token 5 - Position of token 4 ==1 **and** Position of token 4 - Position of token 3 == 1 **and** Position of token 3 - Position of token 2 == 1 **and** Position of token 2 - Position of token 1 ==1 **and** position of token 5 - Position of token 1 == 4.
- **The number of tokens > 5:** Ignore the query.