Question 1-

Methodology-

Creation of unigram index-

1) Read all files one-by-one
2) Preprocessing the files to get the token list
3) Iterate over all the tokens of the token list
4) If the token is present, in the index, add the document is being processed in the posting list of the unigram index.
5) If not present, in an empty list, add the document id.
6) After all the tokens are iterated over, token_file will contain all the tokens along with their doc IDs

User queries for various operations-

1) Firstly, we took the inputs as the number of queries, then the input statement, and then the different operations.
2) Then we preprocessed the given statement and turned it into a list of words.
3) Now we select the first two words and perform the first operation and save the results in a set, which is then used as a list to perform the next operation with the next word.
4) We have performed four operations, namely ‘or’, ‘and’, ‘or not’ and ‘and not’.

Preprocessing-

1) Removing headers and footers
2) Lemmatization on the tokens
3) Converting numbers to string
4) Removing apostrophe
5) Used TweetTokenizer() for tokenizing the string into words
6) Using stopwords, removed all the stopwords such as not and in from the token list.
7) Removed all the punctations from the words in the token list
8) Changed all the words to their lower cases
9) Removed all the blank spaces from the token list

Assumptions-

1) No spelling detection is done on tokens
2) The format would be [op1,op2..]

Question 2
Use the same dataset given in the previous question.
(a) (2.5 marks) Carry out the following preprocessing steps on the given dataset
(i) Convert the text to lower case
(ii) Perform word tokenization
(iii) Remove stopwords from tokens
(iv) Remove punctuation marks from tokens
(v) Remove blank space tokens
(b) (2.5 marks) Implement the positional index data structure
(c) (5 marks) Provide support for the searching of phrase queries. You may assume query length to be
less than or equal to 5.

Preprocessing steps-

1) Used TweetTokenizer() for tokenizing the string into words
2) Using stopwords, removed all the stopwords such as not and in from the token list.
3) Removed all the punctations from the words in the token list
4) Changed all the words to their lower cases
5) Removed all the blank spaces from the token list

Methodology
the positional index-

1) In the case, of the positional index, positing list contains a list of documents where the respective term occurs. It gets sorted by the doc id and is stored in the dictionary.
2) Steps-
   i) fetching the respective document
   ii) preprocessing the document, and getting the token list
   iii)if the word is already present in the dictionary, add that document adn the respective positi0on of the word in the dictionary

iv) If not present, create a new entry
v) Keep uploading the frequency of that word or token, and adding it to the posting list.

User queries-

1) Preprocessing is done on the input query
2) Tokens are collected
3) If the length of the list is 0, then empty is printed
4) If the length of the list is 1 normal position of the word, the documents found are printed along with the frequency.
5) For the dictionary of the first token in form of { doc id, position array}, iterate over all the doc IDs of the first token, and for all positions in that list as start position index and see for the subsequent query tokens. That very doc ID would be the keys and the next start as the starting position.
6) With these conditions, the docid is added to the list.

Assumptions-

1) Did not perform stemming or lemmatization on the list.
2) Input query will have a max size of 5.