Preprocessing strategy:

1. Lowercase
2. Remove Extra Spaces
3. Lemmatize using WordNetLemmatizer()
4. Tokenize using RegexpTokenizer()
5. Remove stopword

Methodology:

In question 1, the processing of queries is handled through various functions such as do_and(), do_or(), etc. These functions process the query by the two-pointer method. Two pointers traverse the postings and compare the entities. Thus the time complexity of these operations in O(x+y) where x and y are the lengths of the posting lists. Further, we execute each operation from the set of operations from left to right where each operation has the complexity O(x+y).

In question 2, for the phrase queries, we divide the phrase into words and get the documents and corresponding positions where the positions must differ by 1. Such indices are acquired for every document id that matches the two posting lists. Finally, we iterate over the list of words in a pairwise fashion by feeding the output of previous pairs into the next.

Assumptions:

1. Only a single phrase query is provided.
2. Lots of garbage characters were throwing errors in UTF-8 encoding, thus we have used windows-1254 encoding and ignored the errors that were arising.

How to Run:

Question 1-

We have written the code in the Jupyter notebook. The first couple of cells needs to be run for getting the unigram inverted index.
Afterwards, the query needs to add as a string and the list of operations as a list of strings. Running the next cell will preprocess the query and generate the output.

Question 2-

Similar to question 1, the first couple of cells need to be run to get the positional index. The phrase query needs to be added as a string and running the next cell will generate the desired output.