# CSE508: Information Retrieval
# Assignment 1

Rishi Raj 2019090
Robin Garg 2019092

Q 1

a. Preprocessing steps (using nltk python library)-
   1. Removing punctuations using regex
   2. Tokenizing the words
   3. Lower case all the tokens
   4. Remove English stopwords

b. Inverted index data structure - We used a dictionary to keep a list of documents in which a token appeared.

c. OR, AND, NOT queries - three separate functions are implemented for each of these queries. AND operation takes the intersection of given two lists and counts the number of comparisons it took. Similarly, OR operation takes the union of two lists and counts the number of comparisons. NOT operation takes the complement of the set of document IDs.

d. Input format

First line - n
Next 2*n lines
Input text
Input operation sequence

```
3
lion stood thoughtfully for a moment
OR, OR, OR
telephone,paved, roads
OR NOT, AND NOT
michael tina
OR

```

Assumptions - input operation sequence is separated by ‘, ‘ (comma and space)
- extra new line at end of input file

Inverted index structure -

For token in a file, add filename to the list corresponding to the token in the dictionary structure

Methodology -

1. Get doc lists of every token.
2. Take not(complement) of list of token next of AND NOT or OR NOT
3. Do all operations of the form x AND y
4. Do all operations of the form x OR y

Order of operations is NOT → AND → OR

Output format (output in python notebook) -

Original given query input
Number of documents retrieved
Number of comparisons done
List of documents

```
Query input = lion OR stood OR thoughtfully OR moment
number of documents retrieved - 185
number of comparisons = 366
['a_tv_t-p.com', 'ambrose.bie', 'anim_lif.txt', 'anime.lif', 'annoy.fascist', 'art-fart.hum', 'b-2.jok', 'barney.txt', 'bbh_intv.txt', 'beauty.t

Query input = telephone OR NOT paved AND NOT roads
number of documents retrieved - 1118
number of comparisons = 2234
['1st_aid.txt', 'a-team', 'a_fish_c.apo', 'a_tv_t-p.com', 'abbott.txt', 'aboutada.txt', 'acetab1.txt', 'aclamt.txt', 'acne1.txt', 'acronym.lis',

Query input = michael OR tina
number of documents retrieved - 110
number of comparisons = 100
['a-team', 'aboutada.txt', 'allfam.epi', 'allusion', 'amazing.epi', 'arnold.txt', 'ateam.epi', 'b12.txt', 'bad-d', 'bitchcar.hum', 'blake7.lis',
```

Q 2

a. First we performed some Preprocessing steps (using nltk python library) as mentioned in the question-
    1. First converted the text to lower case
    2. Then I did word tokenization
    3. Removed stopwords from tokens
    4. Removed punctuation marks from tokens
    5. Removed blank space tokens

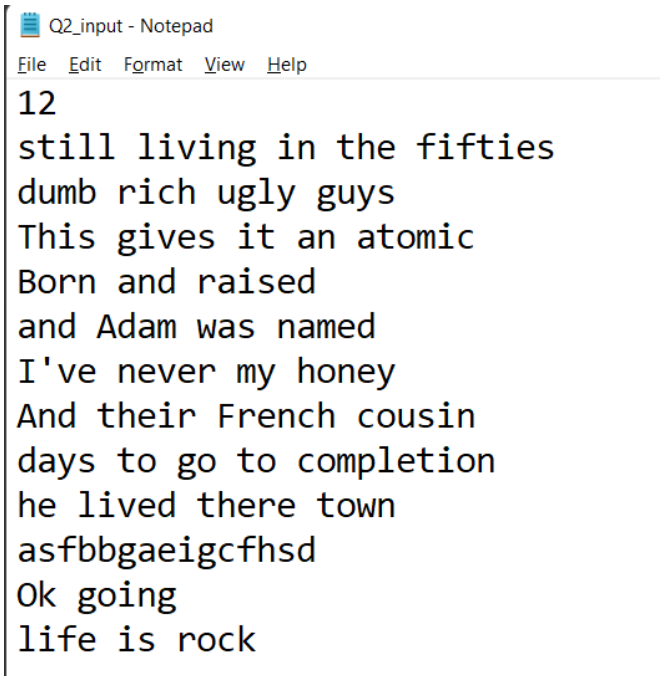
b. I then implemented the positional index data structure where I kept a word dictionary with document IDs and positions of word in the dictionary.

c. For phrase queries, first I performed the same preprocessing steps in the input text, and then I passed it to my query function. Then I first sorted the query using the number of documents the words are present in to make it efficient. After that, I took intersection of all query token to find the phrase query.

INPUT
The Input is given as present in the input file which is first the number of queries and then next n lines contains the phrase to be searched.

OUTPUT
The output is first the number of documents in which the phrase is present and then the list of documents in which the list is present.

Sample Input:




Sample Output:

```
The number of document found for still living in the fifties
 1
The document names are:
acronym.txt

The number of document found for dumb rich ugly guys
 1
The document names are:
acronym.txt

The number of document found for This gives it an atomic
 2
The document names are:
admin.txt
element.jok

The number of document found for Born and raised
 2
The document names are:
aboutada.txt
idaho.txt

The number of document found for and Adam was named
 1
The document names are:
aboutada.txt

I've never my honey
 is not present in any document

The number of document found for And their French cousin
 2
The document names are:
abbott.txt
whoon1st.hum
```

```
The number of document found for days to go to completion
 1
The document names are:
admin.txt

The number of document found for he lived there town
 1
The document names are:
jokes

asfbbgaeigcfhsd
 is not present in any document
The number of document found for Ok going
 2
The document names are:
skippy.hum
skippy.txt

The number of document found for life is rock 1
The document names are:
is_story.txt
```