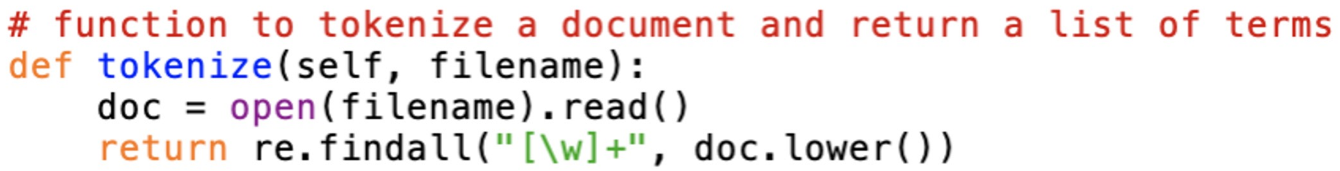
Group Assignment 1
Inverted Index

Team 2: Thanh Le, Antoine Si

IMPLEMENTATION - TOKENIZATION



```
# function to tokenize a document and return a list of terms
def tokenize(self, filename):
    doc = open(filename).read()
    return re.findall("[\w]+", doc.lower())
```

- Open and read the document
- Find and replace all punctuations, numerals, and special characters
- Lowercase all the words in the document

IMPLEMENTATION - BUILDING INDEX

```
# function to read documents from collection, tokenize and build the index with tokens
# index should also contain positional information of the terms in the document --- term: [(ID1,[pos1,pos2,..]), (ID2, [pos1,pos2,…]),….]
# use unique document IDs
def buildIndex(self):
    self.doc_list = {}
    self.index = collections.defaultdict(lambda: collections.defaultdict(list))

    # record the start time
    start_time = time.time()

    # iterate through all documents in the collection
    for filename in os.listdir():
        # check if the document is in text format
        if filename.endswith('.txt'):
            doc_id = len(self.doc_list)

            # add the document into a list
            self.doc_list[doc_id] = filename

            # tokenize the document and build the index with corresponding doc ID and positional information
            for pos, term in enumerate(self.tokenize(filename)):
                self.index[term][doc_id].append(pos)

    # sort the index
    self.index = dict(sorted(self.index.items()))

    # record the end time
    end_time = time.time()

    print("Index built in", end_time - start_time, "seconds.\n")
```

The doc_id of the first document in the collection is 0

The position of the first term in the document is 0

AND MERGE ALGORITHM

Let say we want to merge two posting lists. Following are the steps to be followed to get merge list:

1. Sort the two posting lists using numeric sort by docID
2. Maintain pointers into both posting lists and walk through them simultaneously, in time linear in the total number of postings entries.
3. At each step, compare the docID pointed by both pointers. If they are the same, add that docID into the result list, and advance both pointers. Otherwise, advance the pointer pointing to the smaller docID.
4. Repeat step 3 until it reaches the end of one posting list

If the lengths of the postings lists are m and n, the merge takes O(m + n) operations.

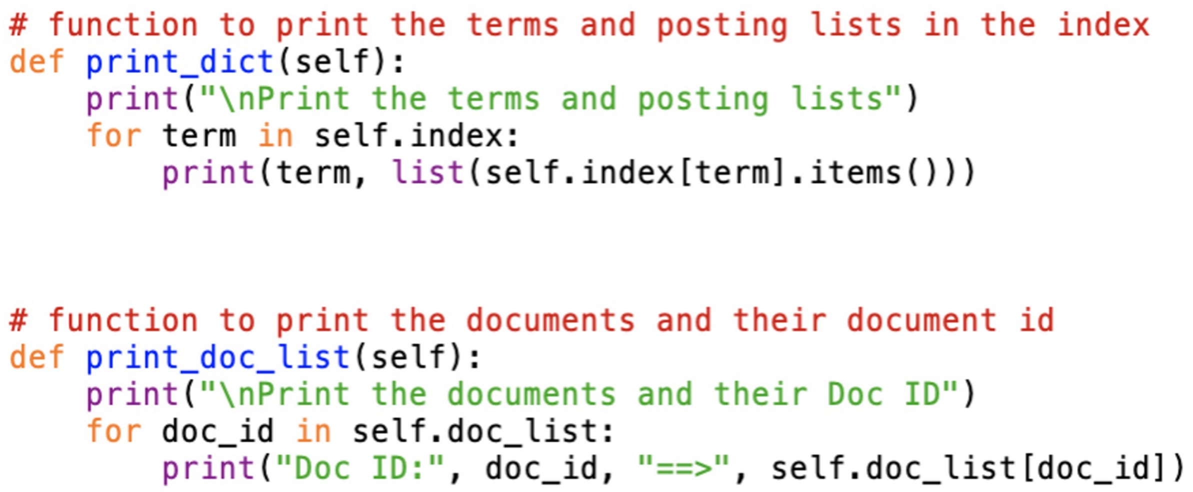
IMPLEMENTATION - AND MERGE

```python
# function to perform "AND merge" on two sorted lists
def and_merge(self, list1, list2):
    result = []

    # construct the pointers
    i, j = 0, 0

    # move the pointers over all entries in the two lists simultaneously until it reaches the end of either one of them
    while i < len(list1) and j < len(list2):
        # if two values being pointed are equal, add the value to the result array and move both pointers
        if list1[i] == list2[j]:
            result.append(list1[i])
            i += 1
            j += 1

        # if the value in list1 is smaller than list2, advance the pointer of list1
        elif list1[i] < list2[j]:
            i += 1

        # if the value in list1 is greater than list2, advance the pointer of list2
        else:
            j += 1

    return result
```

IMPLEMENTATION - AND QUERY

```python
# function to identify relevant docs using the index
def and_query(self, query_terms):
    # record the start time
    start_time = time.time()

    # set the posting lists of the first term in the query as the result lists
    result = [doc_id for doc_id in self.index[query_terms[0]]]

    # iterate through all other terms in the query
    for query_term in query_terms[1:]:
        # get the posting lists of the current term
        docs = [doc_id for doc_id in self.index[query_term]]

        # merge the result lists with the posting lists of the current term
        result = self.and_merge(result, docs)

    # record the end time
    end_time = time.time()

    print("Results for the Query:", " AND ".join(query_terms))
    print("Total Docs retrieved:", len(result))

    for doc_id in result:
        print(self.doc_list[doc_id])

    print("Retrieved in", end_time - start_time, "seconds.\n")
```

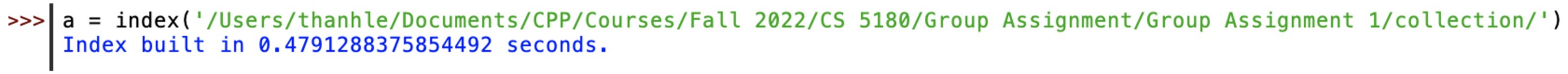
IMPLEMENTATION



```
# function to print the terms and posting lists in the index
def print_dict(self):
    print("\nPrint the terms and posting lists")
    for term in self.index:
        print(term, list(self.index[term].items()))


# function to print the documents and their document id
def print_doc_list(self):
    print("\nPrint the documents and their Doc ID")
    for doc_id in self.doc_list:
        print("Doc ID:", doc_id, "==>", self.doc_list[doc_id])
```

EXPERIMENTAL RESULTS

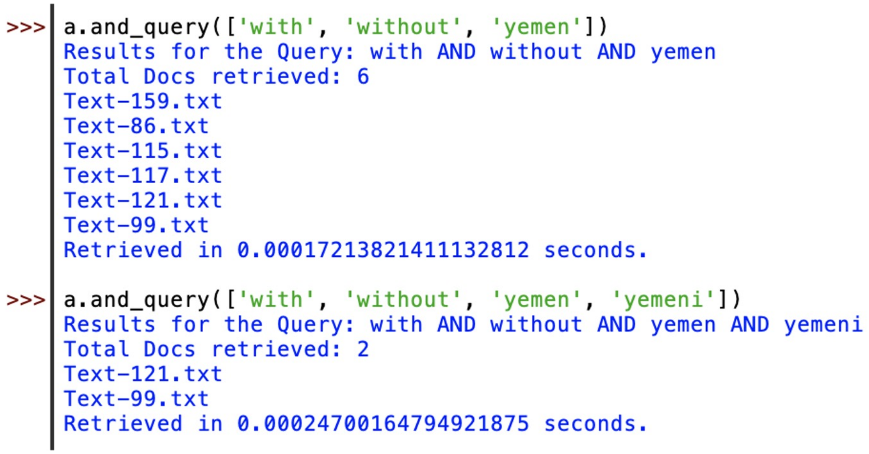
Build the index



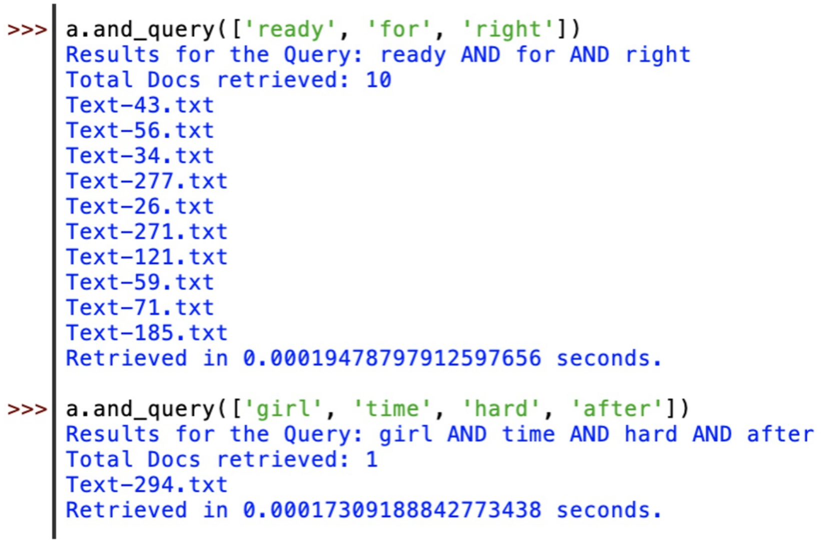
```
>>> a = index('/Users/thanhle/Documents/CPP/Courses/Fall 2022/CS 5180/Group Assignment/Group Assignment 1/collection/')
    Index built in 0.4791288375854492 seconds.
```

EXPERIMENTAL RESULTS

Process the queries



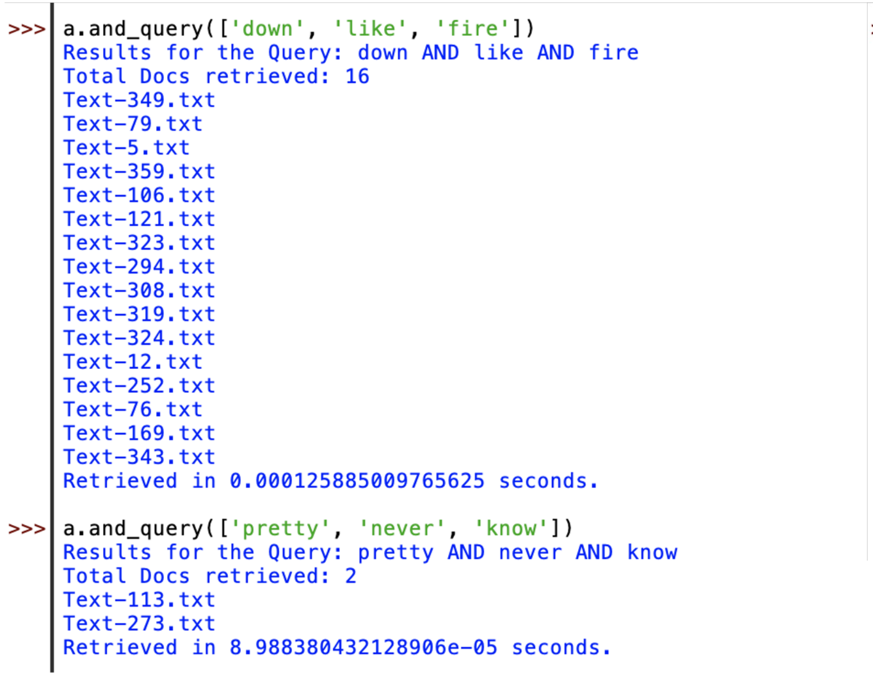
```
>>> a.and_query(['with', 'without', 'yemen'])
Results for the Query: with AND without AND yemen
Total Docs retrieved: 6
Text-159.txt
Text-86.txt
Text-115.txt
Text-117.txt
Text-121.txt
Text-99.txt
Retrieved in 0.00017213821411132812 seconds.

>>> a.and_query(['with', 'without', 'yemen', 'yemeni'])
Results for the Query: with AND without AND yemen AND yemeni
Total Docs retrieved: 2
Text-121.txt
Text-99.txt
Retrieved in 0.00024700164794921875 seconds.
```



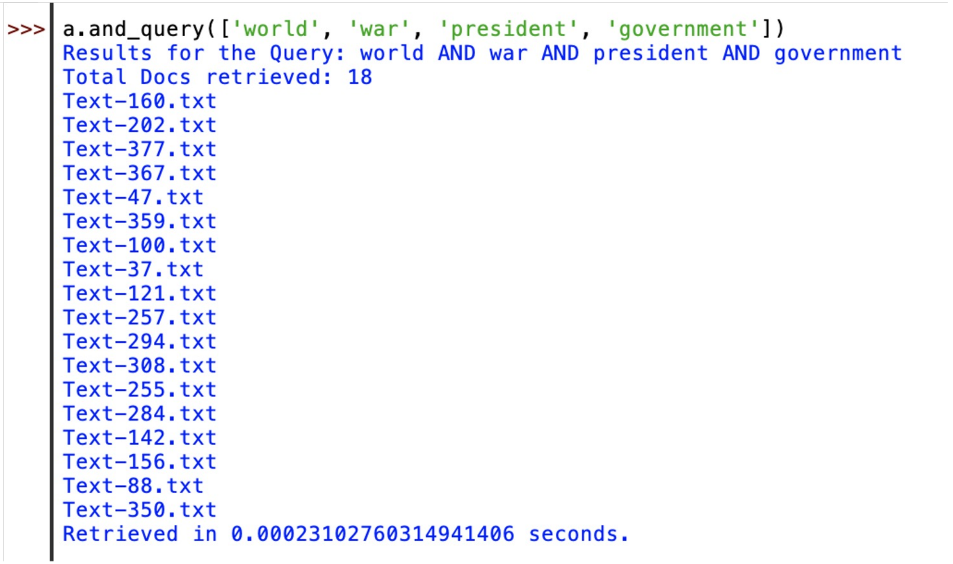
```
>>> a.and_query(['ready', 'for', 'right'])
Results for the Query: ready AND for AND right
Total Docs retrieved: 10
Text-43.txt
Text-56.txt
Text-34.txt
Text-277.txt
Text-26.txt
Text-271.txt
Text-121.txt
Text-59.txt
Text-71.txt
Text-185.txt
Retrieved in 0.00019478797912597656 seconds.

>>> a.and_query(['girl', 'time', 'hard', 'after'])
Results for the Query: girl AND time AND hard AND after
Total Docs retrieved: 1
Text-294.txt
Retrieved in 0.00017309188842773438 seconds.
```

EXPERIMENTAL RESULTS

Process the queries



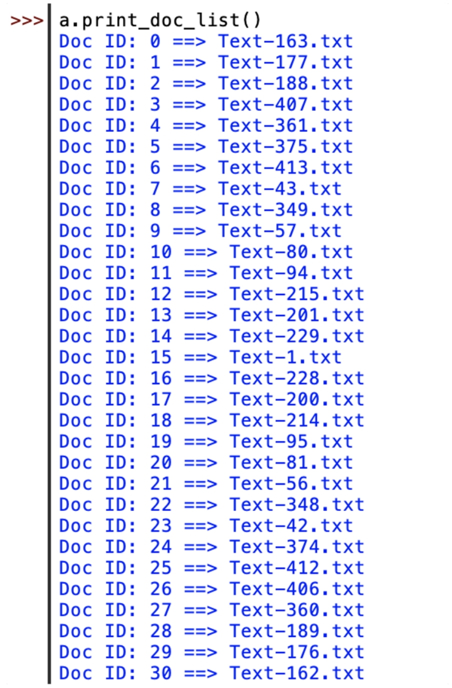
```
>>> a.and_query(['down', 'like', 'fire'])
    Results for the Query: down AND like AND fire
    Total Docs retrieved: 16
    Text-349.txt
    Text-79.txt
    Text-5.txt
    Text-359.txt
    Text-106.txt
    Text-121.txt
    Text-323.txt
    Text-294.txt
    Text-308.txt
    Text-319.txt
    Text-324.txt
    Text-12.txt
    Text-252.txt
    Text-76.txt
    Text-169.txt
    Text-343.txt
    Retrieved in 0.000125885009765625 seconds.

>>> a.and_query(['pretty', 'never', 'know'])
    Results for the Query: pretty AND never AND know
    Total Docs retrieved: 2
    Text-113.txt
    Text-273.txt
    Retrieved in 8.988380432128906e-05 seconds.
```



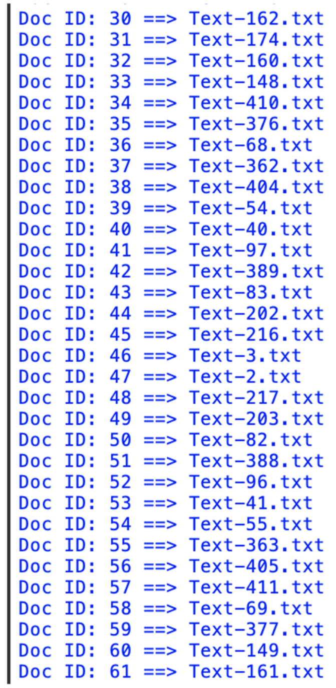
```
>>> a.and_query(['world', 'war', 'president', 'government'])
    Results for the Query: world AND war AND president AND government
    Total Docs retrieved: 18
    Text-160.txt
    Text-202.txt
    Text-377.txt
    Text-367.txt
    Text-47.txt
    Text-359.txt
    Text-100.txt
    Text-37.txt
    Text-121.txt
    Text-257.txt
    Text-294.txt
    Text-308.txt
    Text-255.txt
    Text-284.txt
    Text-142.txt
    Text-156.txt
    Text-88.txt
    Text-350.txt
    Retrieved in 0.00023102760314941406 seconds.
```

EXPERIMENTAL RESULTS

Print out the documents with their Doc IDs

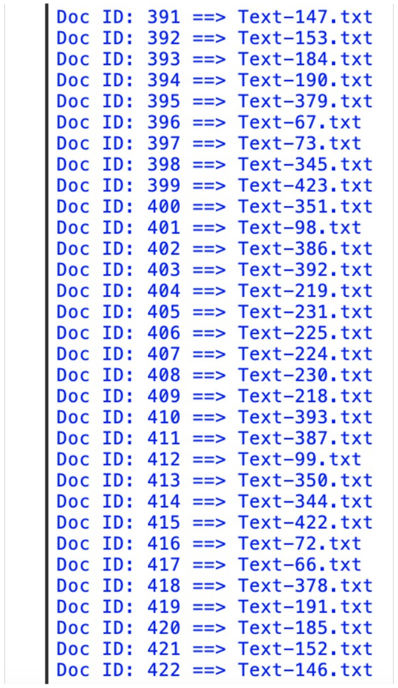


```
>>> a.print_doc_list()
Doc ID: 0 ==> Text-163.txt
Doc ID: 1 ==> Text-177.txt
Doc ID: 2 ==> Text-188.txt
Doc ID: 3 ==> Text-407.txt
Doc ID: 4 ==> Text-361.txt
Doc ID: 5 ==> Text-375.txt
Doc ID: 6 ==> Text-413.txt
Doc ID: 7 ==> Text-43.txt
Doc ID: 8 ==> Text-349.txt
Doc ID: 9 ==> Text-57.txt
Doc ID: 10 ==> Text-80.txt
Doc ID: 11 ==> Text-94.txt
Doc ID: 12 ==> Text-215.txt
Doc ID: 13 ==> Text-201.txt
Doc ID: 14 ==> Text-229.txt
Doc ID: 15 ==> Text-1.txt
Doc ID: 16 ==> Text-228.txt
Doc ID: 17 ==> Text-200.txt
Doc ID: 18 ==> Text-214.txt
Doc ID: 19 ==> Text-95.txt
Doc ID: 20 ==> Text-81.txt
Doc ID: 21 ==> Text-56.txt
Doc ID: 22 ==> Text-348.txt
Doc ID: 23 ==> Text-42.txt
Doc ID: 24 ==> Text-374.txt
Doc ID: 25 ==> Text-412.txt
Doc ID: 26 ==> Text-406.txt
Doc ID: 27 ==> Text-360.txt
Doc ID: 28 ==> Text-189.txt
Doc ID: 29 ==> Text-176.txt
Doc ID: 30 ==> Text-162.txt
```



```
Doc ID: 30 ==> Text-162.txt
Doc ID: 31 ==> Text-174.txt
Doc ID: 32 ==> Text-160.txt
Doc ID: 33 ==> Text-148.txt
Doc ID: 34 ==> Text-410.txt
Doc ID: 35 ==> Text-376.txt
Doc ID: 36 ==> Text-68.txt
Doc ID: 37 ==> Text-362.txt
Doc ID: 38 ==> Text-404.txt
Doc ID: 39 ==> Text-54.txt
Doc ID: 40 ==> Text-40.txt
Doc ID: 41 ==> Text-97.txt
Doc ID: 42 ==> Text-389.txt
Doc ID: 43 ==> Text-83.txt
Doc ID: 44 ==> Text-202.txt
Doc ID: 45 ==> Text-216.txt
Doc ID: 46 ==> Text-3.txt
Doc ID: 47 ==> Text-2.txt
Doc ID: 48 ==> Text-217.txt
Doc ID: 49 ==> Text-203.txt
Doc ID: 50 ==> Text-82.txt
Doc ID: 51 ==> Text-388.txt
Doc ID: 52 ==> Text-96.txt
Doc ID: 53 ==> Text-41.txt
Doc ID: 54 ==> Text-55.txt
Doc ID: 55 ==> Text-363.txt
Doc ID: 56 ==> Text-405.txt
Doc ID: 57 ==> Text-411.txt
Doc ID: 58 ==> Text-69.txt
Doc ID: 59 ==> Text-377.txt
Doc ID: 60 ==> Text-149.txt
Doc ID: 61 ==> Text-161.txt
```

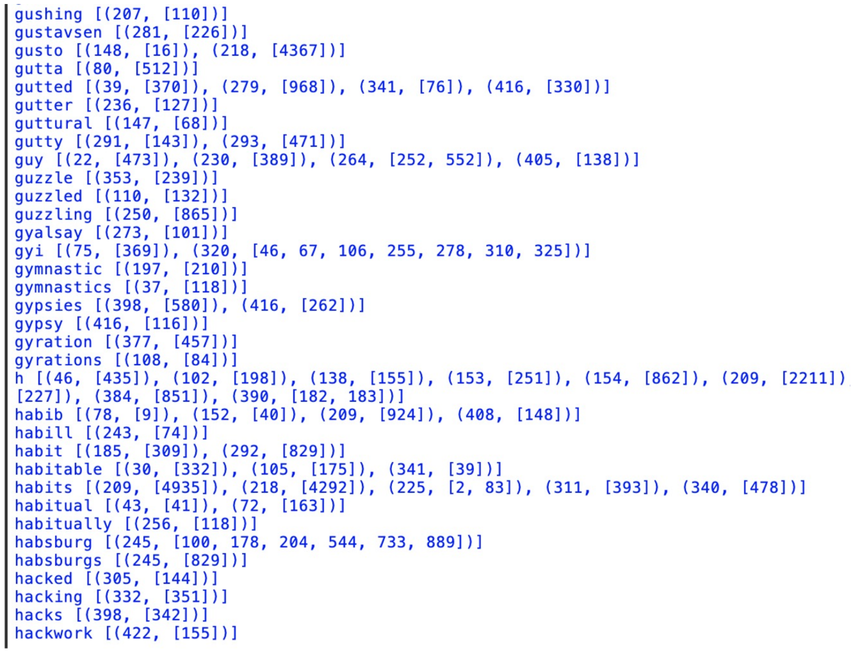
...



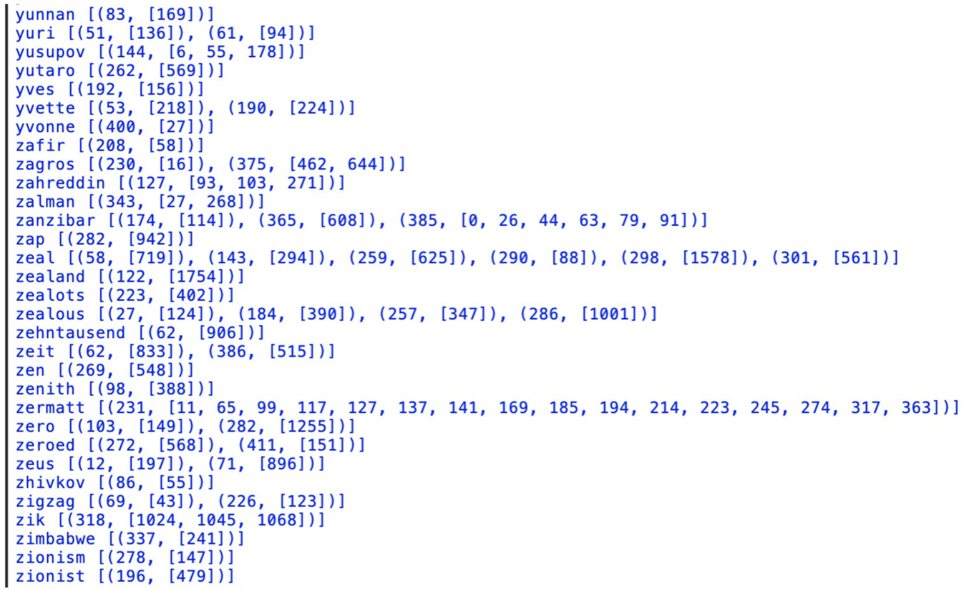
```
Doc ID: 391 ==> Text-147.txt
Doc ID: 392 ==> Text-153.txt
Doc ID: 393 ==> Text-184.txt
Doc ID: 394 ==> Text-190.txt
Doc ID: 395 ==> Text-379.txt
Doc ID: 396 ==> Text-67.txt
Doc ID: 397 ==> Text-73.txt
Doc ID: 398 ==> Text-345.txt
Doc ID: 399 ==> Text-423.txt
Doc ID: 400 ==> Text-351.txt
Doc ID: 401 ==> Text-98.txt
Doc ID: 402 ==> Text-386.txt
Doc ID: 403 ==> Text-392.txt
Doc ID: 404 ==> Text-219.txt
Doc ID: 405 ==> Text-231.txt
Doc ID: 406 ==> Text-225.txt
Doc ID: 407 ==> Text-224.txt
Doc ID: 408 ==> Text-230.txt
Doc ID: 409 ==> Text-218.txt
Doc ID: 410 ==> Text-393.txt
Doc ID: 411 ==> Text-387.txt
Doc ID: 412 ==> Text-99.txt
Doc ID: 413 ==> Text-350.txt
Doc ID: 414 ==> Text-344.txt
Doc ID: 415 ==> Text-422.txt
Doc ID: 416 ==> Text-72.txt
Doc ID: 417 ==> Text-66.txt
Doc ID: 418 ==> Text-378.txt
Doc ID: 419 ==> Text-191.txt
Doc ID: 420 ==> Text-185.txt
Doc ID: 421 ==> Text-152.txt
Doc ID: 422 ==> Text-146.txt
```

EXPERIMENTAL RESULTS

Print out the terms with posting lists



```
gushing [(207, [110])]
gustavsen [(281, [226])]
gusto [(148, [16]), (218, [4367])]
gutta [(80, [512])]
gutted [(39, [370]), (279, [968]), (341, [76]), (416, [330])]
gutter [(236, [127])]
guttural [(147, [68])]
gutty [(291, [143]), (293, [471])]
guy [(22, [473]), (230, [389]), (264, [252, 552]), (405, [138])]
guzzle [(353, [239])]
guzzled [(110, [132])]
guzzling [(250, [865])]
gyalsay [(273, [101])]
gyi [(75, [369]), (320, [46, 67, 106, 255, 278, 310, 325])]
gymnastic [(197, [210])]
gymnastics [(37, [118])]
gypsies [(398, [580]), (416, [262])]
gypsy [(416, [116])]
gyration [(377, [457])]
gyrations [(108, [84])]
h [(46, [435]), (102, [198]), (138, [155]), (153, [251]), (154, [862]), (209, [2211])
[227]), (384, [851]), (390, [182, 183])]
habib [(78, [9]), (152, [40]), (209, [924]), (408, [148])]
habill [(243, [74])]
habit [(185, [309]), (292, [829])]
habitable [(30, [332]), (105, [175]), (341, [39])]
habits [(209, [4935]), (218, [4292]), (225, [2, 83]), (311, [393]), (340, [478])]
habitual [(43, [41]), (72, [163])]
habitually [(256, [118])]
habsburg [(245, [100, 178, 204, 544, 733, 889])]
habsburgs [(245, [829])]
hacked [(305, [144])]
hacking [(332, [351])]
hacks [(398, [342])]
hackwork [(422, [155])]
```



```
yunnan [(83, [169])]
yuri [(51, [136]), (61, [94])]
yusupov [(144, [6, 55, 178])]
yutaro [(262, [569])]
yves [(192, [156])]
yvette [(53, [218]), (190, [224])]
yvonne [(400, [27])]
zafir [(208, [58])]
zagros [(230, [16]), (375, [462, 644])]
zahreddin [(127, [93, 103, 271])]
zalman [(343, [27, 268])]
zanzibar [(174, [114]), (365, [608]), (385, [0, 26, 44, 63, 79, 91])]
zap [(282, [942])]
zeal [(58, [719]), (143, [294]), (259, [625]), (290, [88]), (298, [1578]), (301, [561])]
zealand [(122, [1754])]
zealots [(223, [402])]
zealous [(27, [124]), (184, [390]), (257, [347]), (286, [1001])]
zehntausend [(62, [906])]
zeit [(62, [833]), (386, [515])]
zen [(269, [548])]
zenith [(98, [388])]
zermatt [(231, [11, 65, 99, 117, 127, 137, 141, 169, 185, 194, 214, 223, 245, 274, 317, 363])]
zero [(103, [149]), (282, [1255])]
zeroed [(272, [568]), (411, [151])]
zeus [(12, [197]), (71, [896])]
zhivkov [(86, [55])]
zigzag [(69, [43]), (226, [123])]
zik [(318, [1024, 1045, 1068])]
zimbabwe [(337, [241])]
zionism [(278, [147])]
zionist [(196, [479])]
```

TASK DIVISION

- Both of us first started working on implementing the index.py file by ourselves and discussed with each other when any problem happened
- We compared our query results with each other to make sure we got the same outputs
- We combined our works into a final version with detailed comments in each step which helps explaining the codes
- Thanh created the README file
- Antoine created the output.txt file
- We worked on the presentation slides together