In [11]:

```
def get_not(word):
    a = postings[word][0]
    b = set(range(len(paths)))
    return b.difference(a)

s1 = postings['one'][0]
s2 = postings['nine'][0]
s3 = get_not('exam')

print(s1)
print(s2)
print(s3)

print('one AND nine NOT exam = ', s1 & s2 & s3)
```

```
{21 ,20 ,19 ,18 ,17 ,16 ,14 ,13 ,8 ,7 ,5 ,4 ,3 ,2 ,1 ,0}
{19 ,18 ,17 ,5 ,4 ,2 ,1 ,0}
{20 ,19 ,18 ,17 ,16 ,15 ,14 ,13 ,12 ,11 ,10 ,9 ,8 ,7 ,6 ,5 ,4 ,3 ,2 ,1 ,0}
{one AND nine NOT exam =  {0, 1, 2, 4, 5, 17, 18, 19
```

In [12]:

```
def generate_command_tokens(query):
    query = query.lower()
    tokens = word_tokenize(query)

    commands = []
    query_words = []

    for t in tokens:
        if t not in ['and', 'or', 'not']:
            processed_word = preprocess([t], True)
            print(str(processed_word))
            query_words.append(str(processed_word))
        else:
            commands.append(t)

    return commands, query_words
```

In [13]:

```
def gen_not_tuple(query_words, commands):
    tup = []
    while 'not' in commands:
        i= commands.index('not')
        word = query_words[i]
        word_postings = get_not(word)
        tup.append(word_postings)
        commands.pop(i)
        query_words[i] = i
        print("\nAfter Not Processing: ",commands, query_words)
    return tup
```

In [14]:
```
def binary_operations(query_words, commands, tup):
    a = postings[query_words[0]][0]
    query_words.pop(0)

    for i in range(len(commands)):
        if type(query_words[i]) == int:
            b = tup.pop(0)
        else:
            b = postings[query_words[i]][0]

        if commands[i] == 'and':
            a = a.intersection(b)
        elif commands[i] == 'or':
            q= a.union(b)
        else:
            print('Invaled Command')

    return a
```

In [15]:
```
def execute_query(query):
    commands, query_words = generate_command_tokens(query)
    tup = gen_not_tuple(query_words, commands)
    print('\nCommands: ', commands)
    print('\nQuery Words: ', query_words)
    print('\nTup: ', tup)

    final_set = binary_operations(query_words, commands, tup)
    print('\nFinal Set: ', final_set)

    return final_set
```

In [16]:
```
def print_file(file):
    out_file = open(path[file], 'r', encoding='cp1250')
    out_text = out_file.read()
    print(out_test)
```

```
query = 'exam and resourc'
lists = execute_query(query)
```

:[17] In

```
['exam']
['resourc']

['Commands:  ['and

["['Query Words:  ["['exam']", "['resourc

[]  :Tup

---------------------------------------------------------------------------
(KeyError                                  Traceback (most recent call last
 Anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key,\~
(method, tolerance
:try              2896
(return self._engine.get_loc(key                 2897 <-
:except KeyError              2898

()pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc

()pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashT
()able.get_item

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashT
()able.get_item

"['KeyError: "['exam

:During handling of the above exception, another exception occurred

(KeyError                                  Traceback (most recent call last
<ipython-input-17-6aefbf68fdd1> in <module>
'query = 'exam and resourc 1
(lists = execute_query(query 2 <----

(ipython-input-15-33f5c474835b> in execute_query(query>
(print('\nTup: ', tup     6
 7
(final_set = binary_operations(query_words, commands, tup     8 <----
(print('\nFinal Set: ', final_set     9
 10

ipython-input-14-2ac516893b05> in binary_operations(query_words, commands, tu>
(p
:(def binary_operations(query_words, commands, tup 1
[a = postings[query_words[0]][0     2 <----
(query_words.pop(0     3
 4
:((for i in range(len(commands     5

(Anaconda3\lib\site-packages\pandas\core\frame.py in __getitem__(self, key\~
:if self.columns.nlevels > 1              2978
(return self._getitem_multilevel(key                 2979
```

```
(indexer = self.columns.get_loc(key            2980 <-
:(if is_integer(indexer             2981
[indexer = [indexer                 2982

 Anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key,\~
(method, tolerance
(return self._engine.get_loc(key                2897
:except KeyError             2898
return self._engine.get_loc(self._maybe_cast_indexer(ke                2899 <-
((y
indexer = self.get_indexer([key], method=method, tolerance=tole         2900
(rance
:if indexer.ndim > 1 or indexer.size > 1         2901

()pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc

()pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashT
()able.get_item

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashT
()able.get_item

"['KeyError: "['exam
```

:[ ] In