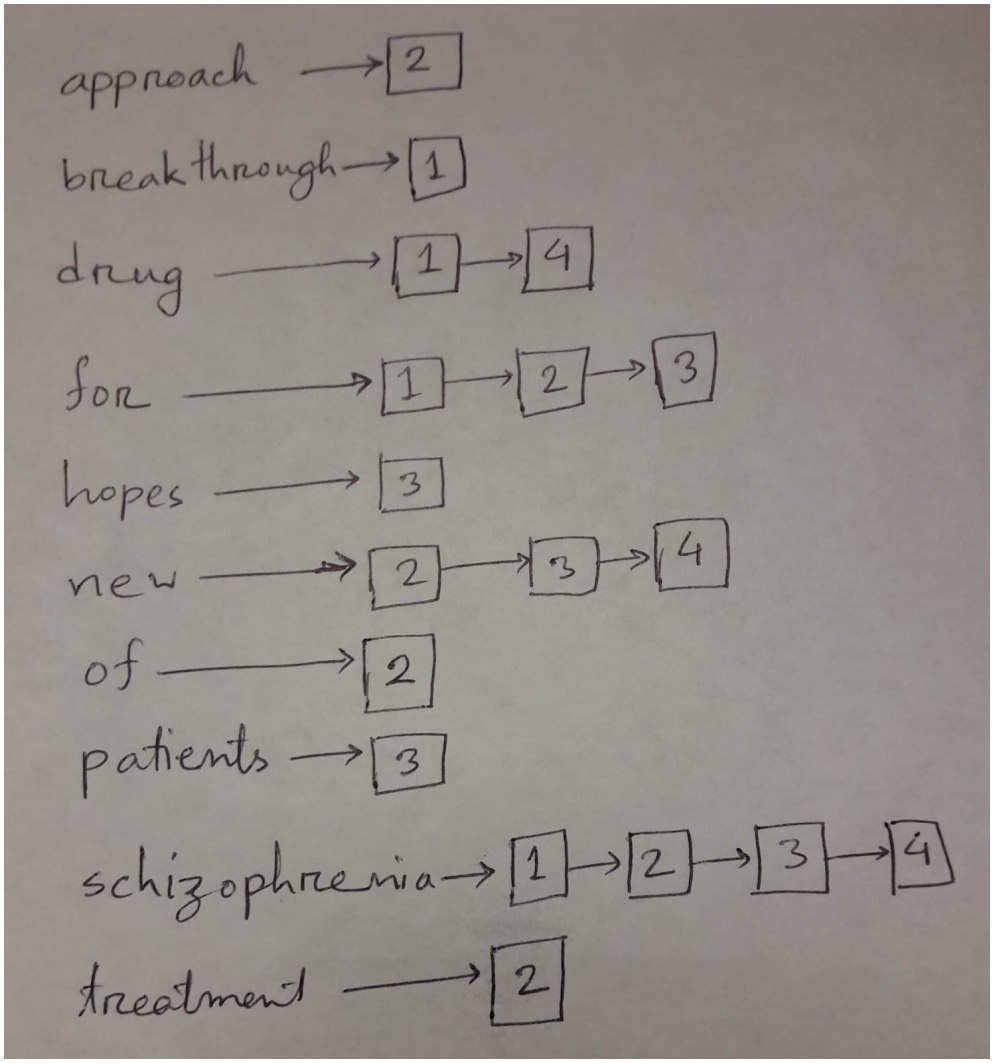
Problem 1

1. Term-document incidence matrix

	Doc 1	Doc 2	Doc 3	Doc 4
approach	0	1	0	0
breakthrough	1	0	0	0
drug	1	0	0	1
for	1	1	1	0
hopes	0	0	1	0
new	0	1	1	1
of	0	1	0	0
patients	0	0	1	0
schizophrenia	1	1	1	1
treatment	0	1	0	0

2. Inverted index representation



3. What are the returned results for these queries:
(a) schizophrenia AND drug
Ans: 1 4
(b) for AND NOT(drug OR approach)
Ans: 3

Problem 2

1.

UNION(Px , Py)
1 answer ← ⟨ ⟩

```
2 while Px ≠ NIL or Py ≠ NIL
3 do  if Px ≠ NIL and Py ≠ NIL
4       If docID( Px ) = docID( Py )
5             then Add(answer , docID( Px ))
6                   Px ← next( Px )
7                   Py ← next( Py )
8             else if docID(Px ) < docID( Py )
9                   then Add(answer , docID( Px )
10                        Px ← next( Px )
11                  else Add(answer , docID( Py )
12                        Py ← next( Py )
13    else if Px ≠ NIL
14            then Add(answer , docID( Px ))
15                  Px ← next( Px )
16            else Add(answer , docID( Py )
17                  Py ← next( Py )
18 return answer
```

2.

```
DIFFERENCE ( Px , Py )
1 answer ← ⟨ ⟩
2 while Px ≠ NIL
3 do  if Py ≠ NIL
4       If docID( Px ) = docID( Py )
5             then  Px ← next( Px )
6                   Py ← next( Py )
7             else if docID( Px ) < docID( Py )
8                   then Add(answer , docID( Px )
9                         Px ← next( Px )
10                  else Py ← next( Py )
11     else Add(answer , docID( Px ))
12                  Px ← next( Px )
13 return answer
```

Problem 3

Recommend a query processing order for:
(tangerine OR trees) AND (marmalade OR skies) AND (kaleidoscope OR eyes)

Answer:
Complexity for (tangerine OR trees) = 87,009 + 316,812 = 403,821

Complexity for (marmalade OR skies) = 107,913 + 271,658 = 379,571
Complexity for (kaleidoscope OR eyes) = 46,653 + 213,312 = 259,965

(kaleidoscope OR eyes) AND (marmalade OR skies) AND (tangerine OR trees)

Problem 4

How should the Boolean query x OR NOT y be handled? Why is the naive evaluation of this query normally very expensive? Write out a postings merge algorithm that evaluates this query efficiently.

Answer:
(x OR NOT y)
= NOT(NOT (x OR NOT y))
= NOT (NOT x AND y)

The Boolean query (x OR NOT y) should be handled by the query (NOT (NOT x AND y)).

Let *d* be the set of the index of all documents and |d| is the number of elements in d. The naive evaluation first call DIFFERENCE (P1 , P2) to find NOT y (= d-y) with complexity =O(|d|) and then call UNION(P1 , P2) to evaluate the query. The loop of UNION terminates when both of the lists are empty. Here the list of (NOT y) contains |d|-|y| elements which is nearly |d|. So, the complexity of naive evaluation is |d|+|d|.
The later representation of the query (NOT (NOT x AND y)) . To find (NOT x AND y), we can just modify the function “Intersect” using same number of loops and having complexity |y| . Then evaluate the whole query by calling DIFFERENCE (P1 , P2) of complexity |d|. The resultant complexity is |d|+|y| . As, in practice, |y| is much less than |d|, the naive evaluation is much expensive.

```
IntersectModified(p1 , p2 )          //p1 is for x, p2 is for y to evaluate NOT x AND y
1 answer ← ⟨ ⟩
2 while p2 ≠ nil
3 do if docID(p1 ) = docID(p2 )
4       then    p1 ← next(p1 )
5               p2 ← next(p2 )
6       else if docID(p1 ) < docID(p2 )
7               then p1 ← next(p1 )
8       else p2 ← next(p2 )
9            Add(answer , docID(p2 ))
10 return answer
```

merge(p1 , p2) // this function evaluates the query
1 DIFFERENCE (p_d , IntersectModified(p1 , p2)) //p1 is for x, p2 is for y, p_d is for d

Problem 5

1. What does your code return for the file above and the query: schizophrenia AND drug?

Answer:

Doc 1
Doc 4

2. What does your code return for the query: breakthrough OR new?

Answer:

Doc 1
Doc 2
Doc 3
Doc 4

3. What does your code return for this query: drug OR treatment AND schizophrenia ?

Answer:

Doc 1
Doc 2
Doc 4