Problem 1

Part 1

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

Part 2

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

Part 3

Part a

This query will yield the following documents: Doc 1, Doc 2

Part b

Let's break this query down. The subquery "(drug OR approach)" will yield documents 1, 2, and 3. The subquery "for" will yield documents 1, 2, and 4. Applying the "AND NOT" operator to these results will leave us with only document 4.

Thus, the ultimate result is document 4.

Problem 2

Part 1

```
def or_operator(term1_documents, term2—documents):
    # Create a set out of the longer list of documents
    term1_length = len(term1_documents)
    term2_length = len(term2_documents)
    if term1_length > term2_length:
        results = set(term1_documents)
    else:
        results = set(term2_documents)

    # Add the smaller collection to the set
    if term1_length > term2_length:
        for document in term2_documents:
            if document not in documents:
                results.add(document)
    else:
        for document in term1_documents:
            if document not in documents:
                results.add(document)

    return results
```

Part 2

```
def not_operator(term1_documents, term2_documents):
    results = []
    for document in term1_documents:
        if document not in term2_documents:
            results.add(document)

    return results
```

Problem 3

Let's first compare the sizes of the resulting postings lists for each of the sub-queries:

- (tangerine OR trees) will yield a postings list containing, at most, 363,465 documents

- (marmalade OR skies) will yield a postings list containing, at most, 379,571 documents
- (kaleidoscope OR eyes) will yield a postings list containing, at most, 300,321 documents

After processing each of these subqueries, we should start by computing the intersection (AND) between the second and third subquery, as that will yield at most 300,321 documents. We should then compute the intersection between the resulting document list and the results of the first subquery.

Thus, the order that the queries would be processed in is:

1. Process each of the the 3 subqueries, (tangerine OR trees), (marmalade OR skies), and (kaleidoscope OR eyes) in any order.
2. Compute the AND of the results from (marmalade OR skies) and (kaleidoscope OR eyes).
3. Compute the AND of the results from (kaleidoscope OR eyes) and the result of the previous AND

Problem 4

The way that we should handle this query is by taking the entire list of documents and filtering those out that occur in either the document list for the operand terms. The naive way to handle this query would be to invert the second documents list and then apply the OR. We can handle this instead by observing the following logical equivalence:

$$x \vee (\neg y) \equiv \neg(\neg x \wedge y)$$

We will first compute the documents associated with the second term that are not associated with the second term. After that, we will filter those documents from the global documents list. In code, that might look like this:

```
def or_not_operator(term1_docs, term2_docs, all_docs):
    # Calculate items we don't want
    not_present = []
    for document in term2_docs:
        if document not in term1_docs:
            not_present.append(document)

    # Calculate the final list
    results = all_docs
    for document in not_present:
        results.remove(document)

    return results
```