Exercise 1

The answer is no. For a conjunctive query, it's likely that intersecting (or merging) the postings lists in order of size will give the fastest processing time, but it is not always guaranteed to be optimal. Suppose that we have 3 postings lists (A, B, and C), where C is the list with the smallest size, and we are attempting to do the intersection $A \cap B \cap C$. An example where this method is not optimal is when:

- $(A \cap B)$ gives the best case scenario, when $(A \cap B) = \emptyset$ (i.e. A and B have no common documents), and
- $(B \cap C)$ gives the worst case scenario, when $(B \cap C) = C$ (i.e. the size of the intermediate result is equal to the size of the postings list C itself).

To be more concrete, let:

A:	9	10	11	12	13	14	15	16
B:	1	2	3	4	5	6	7	8
C:	1	2	3	4	5			

Doing the intersection in order of size, such that $A \cap (B \cap C)$, will perform 10 comparisons (twice the size of C). On the other hand, if we perform $(A \cap B) \cap C$, then we only need 8 comparisons, because $(A \cap B)$ will return an empty set, and thus will terminate the computation right way. The difference between 8 and 10 might seem small, but for larger postings lists, the difference will be quite significant. Besides, this example proves that performing the intersection in order of size is not always the optimal solution.

Exercise 2

Skip pointers are often used to skip through the documents that are *not* common between the two postings lists that we are intersecting. However, for queries of the form x OR y, we are interested in the union of the two lists (i.e. the documents that are *not* common between the lists are also important). Hence, skip pointers are not useful for these cases.

Exercise 3

(a) For query Gates /2 Microsoft, we will get document 1 and 3 as the result.

(b) For $k = 1$, we get $\emptyset$ (i.e. no documents have the word Gates and Microsoft standing next to each other).

For $2 \leq k < 5$, we get $\{1,3\}$.

For $5 \leq k$, we get $\{1,2,3\}$.

Exercise 4

(a) *In a Boolean retrieval system, stemming never lowers precision.* This is **false**. Stemming is a technique that reduces a word to its base form, regardless of what context the word is in. For instance, the words "worse", "good", and "better" will all be mapped to the base word "good" after the stemming process. By doing this, we lose a bit of information, leading to a larger number of retrieved documents that are not relevant to the user's information need. Suppose that the user wants to search for "Google is [better] than Bing", then due to stemming, not only "better", but also "worse" and "as good as" will be returned as the results, which are not really what the user wants.

(b) *In a Boolean retrieval system, stemming never lowers recall*. This is **true**. Recall is defined as the fraction of documents in the collection that are retrieved. As I have already mentioned in (a), stemming results in more documents being retrieved. Thus stemming actually increases recall.

(c) *Stemming increases the size of the vocabulary*. This is **false**. Stemming can be viewed as a way to group several words into one specific base form. This means that there are less unique words in our vocabulary compared to when stemming is not used. Hence, stemming should decrease the size of the vocabulary, not increasing it.

(d) *Stemming should be invoked at indexing time but not while processing the query*. This is **false**. Stemming is a reduction technique, and thus must be invoked both during the indexing time as well as the query time, such that the lookup can be "symmetric". If stemming is only invoked at indexing time, we will not be able to find any results for non-stemmed words (e.g. worse or better), because our inverted index only contains the stemmed word (e.g. good). Hence, we must always use stemming both during indexing time as well as in the queries.

On paper (Optional for INF3800)

Exercise 1:

For a conjunctive query, is processing postings lists in order of size guaranteed to be optimal? Explain why it is, or give an example where it isn't.

(Exercise 1.9, page 13)

Exercise 2:

Why are skip pointers not useful for queries of the form x OR y?

(Exercise 2.5, page 38)

Exercise 3:

Consider the following fragment of a positional index with the format:

word: document: <position, position, ...>; document: <position, …>; ...

Gates: 1: <3>; 2: <6>; 3: <2,17>; 4: <1>;
IBM: 4: <3>; 7: <14>;
Microsoft: 1: <1>; 2: <1,21>; 3: <3>; 5: <16,22,51>;

The /k operator, word1 /k word2 finds occurrences of word1 within k words of word2 (on either side), where k is a positive integer argument. Thus k = 1 demands that word1 be adjacent to word2.

a. Describe the set of documents that satisfy the query Gates /2 Microsoft.
b. Describe each set of values for k for which the query Gates /k Microsoft returns a different set of documents as the answer

(Exercise 2.10, page 44)

Exercise 4:

Stemming:
Are the following statements true or false? Justify your answers.

a) In Boolean retrieval system, stemming never lowers precision.
b) In Boolean retrieval system, stemming never lowers recall.
c) Stemming increases the size of the vocabulary.
d) Stemming should be invoked at indexing time but not while processing the query.

(This task is taken from the V15 final exam)