Problem 1

Part 1

The results of the query ”information retrieval” are: Document 1(0.509492), Document 2(0.509492), Document 3(0.095837)

Part 2

The default model for scoring documents in Lucene is the tf-idf scoring model based on the following formula:

$$\text{cosine-similarity}(query, document) = \frac{V(query) \cdot V(document)}{|V(query)| \cdot |V(document)|}$$

where $V(query)$ and $V(document)$ denote the weighted query and document vectors respectively. This is complicated by the addition of things such as ”query boosting” whereby a user can boost the importance of a term in a Lucene query with the ”ˆ” operator.

In addition to using the tf-idf model, the boolean query model is used before scoring occurs to limit the number of documents that must be scored by the system and to handle boolean logical operators that are found in the query such as AND, OR, and NOT.

Part 3

Part a

The results of the query ”information AND retrieval” are: Document 1(0.509492), Document 2(0.509492)

Part b

The results of the query ”information AND NOT retrieval” is: Document 3(0.312500)

Part c

The results of the query ”information AND retrieval WITHIN 1 WORD OF EACH OTHER” is: Document 1(0.714901)

Problem 2

First we should convert the entries that we can to gaps. This yields the list: 777, 16966, 276325, 30975268. Next, we should convert these to binary:

$$\begin{gathered}777 \rightarrow 00000011\ 00001001 \\ 16966 \rightarrow 01000010\ 01000110 \\ 276325 \rightarrow 00000100\ 00110111\ 01100101 \\ 30975268 \rightarrow 00000001\ 11011000\ 10100101\ 00100100\end{gathered}$$

Variable Byte Encoding

We will dedicate the first bit (leftmost) of each 8-bit block to be the continuation bit. Thus, we can encode the numbers like so:
00000110 10000100, 00000010 00000100 11000110, 00010000 01101110 11100101, 00000111 00110001 00100101 1100100

Gamma Codes

777 = 1111111110100001001
16966 = 111111111111111000001001000110
276325 = 11111111111111111110000011011101100101
30972568 = 11111111111111111111111110110110001010010100100100

So the final encoding is: 11111111 10100001 00111111 11111111 10000010 01000110 11111111 11111111 11000001 10111011 00101111 11111111 11111111 11111011 01100010 10010100 100100

Problem 3

Parsed Gamma coding: 1001 110 11 1110111 11 1.
Gaps: 9, 6, 3, 119, 3, 1
Doc Ids: 9, 15, 18, 137, 140, 141

Problem 4

	Doc1	Doc2	Doc3
car	4.01	2.64	2.93
auto	3.07	5.24	2.08
insurance	1.62	4.08	3.99
best	3.22	1.50	3.35

Problem 5

Part 1

First, consider the formula for idf_t, where t is some term is defined as:

$$idf_t = log_{10}\left(\frac{N}{df_t}\right)$$

where df_t is the document frequency of t and N is the number of documents in the collection.

For a term t that occurs in every document, its document frequency would be equal to N. In this case, the fraction $\frac{N}{df_t}$ simply reduces to 1 and the idf_t is the same as $log_{10}(1)$, which is just 0. This would make its tf-idf weight 0, meaning it would have no effect on the ranking of different documents in the collection. This mimics the behavior of a stop word list, which seeks to ignore certain words that add nothing to a documents usefulness (the, is, etc.).

While these are functionally equivalent as far as queries are concerned, handling useless words at indexing time would likely prove to be more efficient in the long run as it would reduce the number of weights that you need to compute for queries.

Part 2

Rewriting $Score(q, d)$ in terms of the *idf* definition:

$$\sum_{t \in q} \left(tf \cdot log_b \left(\frac{N}{df_t} \right) \right)$$

where b is usually considered to be 10. We can rewrite this formula in terms of logarithms with base 10:

$$\sum_{t \in q} \left(tf \cdot \frac{log_{10} \left(\frac{N}{df_t} \right)}{log_{10}(b)} \right)$$

Notice that the logarithms that involve b are all in the denominators of their respective functions. Since logarithm is a monotonically increasing function, we know that as b gets larger, the result of $log_{10}(b)$ will get larger. This means that the $tf - idf$ weights will all decrease. While all $tf - idf$ weights will decrease, they will still retain the same relative ordering and the results of queries will be the same.