Questions:
You will need to use the Lucene API and/or Lucene source code for answering some of these questions.
1. Does the commandline demo system use stopwords? How do you know? If it uses stopwords, what stopwords does it use (list them)?

Yes, you could either demonstrate this empirically by showing a search for a stop word like 'the' returns no hit even though it appears in many of the documents or you could say where in the code stop word are being used. The stop word list is found in StopAnalyzer and consists of the following:
"a", "an", "and", "are", "as", "at", "be", "but", "by",
"for", "if", "in", "into", "is", "it",
"no", "not", "of", "on", "or", "such",
"that", "the", "their", "then", "there", "these",
"they", "this", "to", "was", "will", "with"

2. Does the commandline demo system do stemming? How do you know?

No, this can be observed by entering 'love' and 'loves' as search terms. If the program was stemming we would expect these to return the same results. Since they don't we can assume the program is not stemming. Inspection of the code also reveals no stemming.

3. How does Lucene make searches case insensitive (in the demo)?

In the StandardAnalyzer used by both the indexer and the searcher the token stream is passed through the LowerCaseFilter which goes through the tokens setting each character within the token to lower case.

4. What does Lucene do with apostrophes (in the demo)? How does it do it? (Your answer should state what is done, by what method, of what class).

In the StandardFilter which is used by the StandardAnalyzer we see that when incrementToken() is called, it tests for and removes 's from words. This is the answer I was looking for and expected. However, when looking deeper we see that apostrophes are also removed from the leading and trailing edges of words. This trimming takes place in the StandardTokenizerImpl.

5. Submit the following queries to your search engine and report the hits by listing them in order. (Note: this is to be done before you do steps 6 and 7).

a. about the prince
RJ1.txt
RJ10.txt
RJ2.txt
b. his fiery sword
RJ3.txt
RJ5.txt
RJ4.txt
RJ9.txt
RJ6.txt

c. *ALAS O Love*
RJ6.txt
RJ8.txt
RJ7.txt
RJ1.txt
RJ10.txt
d. love
RJ6.txt
RJ7.txt
RJ8.txt
RJ1.txt
RJ10.txt
e. loves
No matching documents

6. Why do you think the hit order is different between queries *c* and *d*? Speculation is fine; you don't have to figure out the Lucene scoring algorithm at this point.

'Love' appears more times in 7 then it does in 8 which is why it's ranked higher in the second query. However, 'O' appears in 8 where it does not in 7. So it seems that the ranking algorithm weights matches of more terms in the query, 'O' and 'love', higher than the frequency of terms in a document.

7. Submit the following queries to your search engine and report the hits by listing them in order. (Note: this is to be done **after** you do steps 6 and 7 above).
a. about the prince
RJ3.txt
RJ2.txt
b. his fiery sword
RJ3.txt
c. alas o love
RJ6.txt
RJ8.txt
RJ7.txt
RJ1.txt
RJ10.txt
d. loves
RJ6.txt
RJ7.txt
RJ8.txt
RJ1.txt
RJ10.txt

8. Does it matter whether stemming occurs before or after stopword removal? (Consider this as a general question, not just with respect to these documents or stopwords). Why?

Yes, Say we made 'lov' a stop word. If we stem after we remove stop words the word "loves" will be still be indexed as 'lov'; whereas, if we stem prior to the removal of stop words "loves" will not be indexed. So it is important to know when defining stop words if and when stemming is occurring so that the correct words will be removed from the index.

9. What does the Lucene *Field* class do? A one or two sentence summary is all that is necessary.

The Lucene Field class is used to create sections within documents. For instance you could define a "title" field in which you place the contents of the title of a document. This subdivision of documents allows us to make decisions on how to handle the content based on the field where it is found. For instance we may choose to weight the title section more heavily than the rest of the text or we could not index it all together.

10. How might you (programmatically) get the frequency of a particular term in a particular document that has been indexed by Lucene? What method of what class could you use?

To do this programmatically first you need to access the index using IndexReader. Then you can call the method termDocs(Term term) with the term for which you want the frequency. This returns an enumeration of TermDocs all of which contain the term you searched for. You can run through this enumeration until you find the document you're interested in. Then you can call the freq() method to find out the frequency of the term within that document. This is not the only way to accomplish the task.