Assign

Pen and Paper Task 1

A) 1.) No Tokenization and preprocessing necessary

2.) Generate postings:

term	doc ID
he	1
is	1
he	1
is	1
he	1
is	1
he	1
is	1
he	1
is	1
nice	1
he	2
she	2
she	3
nice	3
he	4
he	5
she	6
she	7
she	8
he	9
nice	9

3.) sort postings:

term	doc ID
he	1
he	1
he	1
he	1
he	1
he	2
he	4
he	5
he	9
is	1
is	1
is	1
is	1
is	1
is	1
nice	1
nice	3
nice	9
she	2
she	3
she	6
she	7
she	8

4) Create postings list + determine doc. frequency

term	doc. freq.	→ postings lists
he	5	1 → 2 → 4 → 5 → 9
is	1	1
nice	3	1 → 3 → 9
she	5	2 → 3 → 6 → 7 → 8

B) postings list with skip pointers:

term	doc. freq.	→ post. list
he	5	1 → 2 → 4 → 5 → 9
is	1	1
nice	3	1 → 3 → 9
she	5	2 → 3 → 6 → 7 → 8

Ex. query: "he" AND "nice"

So first we start with both by (1) and that they're the same, so we move both forward. Now we're at (2) and (3), since (3) > (2), we go from (2) to (4). Now (4) > (3), so we go from (3) to (9). So now we can use the skip pointer because (9) > (4) and (9) ≤ (9), so we do not have to look at (5).

Pen and Paper Task 2:

No, because we have to check every doc ID. Since OR is true even if only one of the ~~t~~ is in the document.

Pen and Paper Task 3:

- Strings ~~tok~~ that are stored in the permterm index for car:
  car$, ar$c, r$ca, $car
- How is the permuted index queried for c*r?
  c*r → lookup r$c*
  So the answer would be r$ca

Pen and Paper Task 7:

This doesn't work well because there could be answers where no "v" is included. This could be addressed by searching for re*(and then do a postfiltering where all answers are not included that ~~contain~~ answers to the query re*v AND v*(.