InvertedIndex.java

```
65      while (itr.hasMoreTokens()) {
66          term = itr.nextToken(); // fetch a term in the document
67          if (wordcount.containsKey(term)) {
68        // we've seen this term before, so the term is already in the hash table and we should increase its count by one
69        //#########################################################//
70        // Add one missing statement here to increase the counter by one
71        // Hint:  wordcount.put(???)
72        //#########################################################//
73        wordcount.put(term, wordcount.get(term)+1);
74          } else {
75        // this is first time we see this word, set value '1'
76        //#########################################################//
77        // Add one missing statement here to set the counter to 1
78        // Hint: wordcount.put(???)
79        //#########################################################//
80        wordcount.put(term, 1);
81          }
82      }
83      for (String s : wordcount.keySet()) {
84          word.set(s);
85          // how to set the value for "did" appropriately so that the emitted pair "word" and "did" would
86          // represent what we'd like to pass to the Reducer?
87          //#########################################################//
88          // Add one missing statement here to set the right value for "did"
89          // Hint:  did.set(???)
90          //#########################################################//
91          did.set(docID+" "+wordcount.get(s).toString());
92          output.collect(word, did);
93      }
```

```
110     while (values.hasNext()) { // what should we do for each value fecthed?
111         //#################################################################//
112         // add one line to use "sum" to form a concatenation of all the values
113         // Hint: use "values.next().toString()" to obtain the value
114         //
115         //#################################################################//
116         sum = sum + values.next().toString() + " ";
117     }
```

IndexGeneration.java

```
68
69          while (st.hasMoreTokens()) {
70              // iterate over all the (docID count) pairs and copy them to foutposting.
71              // first, copy the docID using foutposting.writeUTF.
72
73              foutposting.writeUTF(st.nextToken());
74
75              if (st.hasMoreTokens()) {
76              // we should expect another token for the term frequency/count
77              freq = Integer.parseInt(st.nextToken().trim());
78              //#########################################################//
79              // add a statement here so that in the end of the loop "count" would have the total
80              // count of the term in all the documents
81              // Hint: how to update "count"?
82              //
83              //#########################################################//
84              count = count + freq;
85              foutposting.writeInt(freq); // copy the term frequency/count to foutposting
86              } else {
87              System.err.println("Term frequency is expected");
88              }
89
90              //#########################################################//
91              // add a statement here to use "df" to count how many documents contain the term
92              // Hint: how to update "df"?
93              //
94              //#########################################################//
95              df = df + 1;
96          }
```

Retrieval.java

```
160             //////////////////////////////////////////////////////////////////
161             // add a statement here so that after the loop, totalTermCount would
162             // have the total count of the term in the whole collection
163             // Hint: how to update "totalTermCount"?
164             //
165             //////////////////////////////////////////////////////////////////
166             totalTermCount = totalTermCount + termCount;
220               if (s1 != null) {
221                   // this means that the docID already has an entry in the accumulator, i.
222
223                   //////////////////////////////////////////////////////////////////
224                   // add one statement here to update the accumulator
225                   // Hint: (1) use the function "acc.put(docID, ... )"; (2) use "s1.double
226                   //       to obtain the actual score value in s1.
227                   //
228                   //////////////////////////////////////////////////////////////////
229                   acc.put(docID, tmpWeight+s1.doubleValue());
230
231               } else {
232                   // otherwise, we need to add a score accumulator for this docID and set
233                   //////////////////////////////////////////////////////////////////
234                   // add one statement here to set the score accumulator to the right valu
235                   // Hint: (1) use the function "acc.put(docID, ... )"
236                   //
237                   //////////////////////////////////////////////////////////////////
238                   acc.put(docID, tmpWeight);
239               }
```

result:

```
Set average over 19 topics
Set average (non-interpolated) precision = 0.0423747959322594
Set total number of relevant docs = 1809
Set total number of retrieved relevant docs = 593
Set average interpolated precision at recalls:
 avg prec at 0 = 0.287273505384425
 avg prec at 0.1 = 0.0904499800891735
 avg prec at 0.2 = 0.067582294424398
 avg prec at 0.3 = 0.0548373885237802
 avg prec at 0.4 = 0.0439550932981308
 avg prec at 0.5 = 0.0372552742353846
 avg prec at 0.6 = 0.03174512779855
 avg prec at 0.7 = 0.0232357796343913
 avg prec at 0.8 = 0
 avg prec at 0.9 = 0
 avg prec at 1 = 0
Set non-interpolated precsion at docs:
 avg prec at 5 = 0.136842105263158
 avg prec at 10 = 0.126315789473684
 avg prec at 15 = 0.108771929824561
 avg prec at 20 = 0.0973684210526316
 avg prec at 30 = 0.0929824561403509
 avg prec at 100 = 0.0668421052631579
 avg prec at 200 = 0.053421052631579
 avg prec at 500 = 0.0365263157894737
 avg prec at 1000 = 0.0237894736842105
Set breakeven precision = 0.0658398456486486
"pr" 559L, 13887C
```