Result:

```
europe:
186 0.6421503023473478
62 0.6006766055492684
199 0.6006766055492684
173 0.5663233347786729

stock rally:
190 0.6875022275576632
8 0.636503380911287
102 0.636503380911287
145 0.636503380911287
165 0.636503380911287
26 0.5498236888693394
82 0.5498236888693394
38 0.509037822486324
186 0.509037822486324
18 0.1581941341337241

debt crisis:
127 0.7071067811865475
96 0.6666666666666666
68 0.6055217603514403
98 0.6055217603514403
121 0.6055217603514403
78 0.5664137418678318
136 0.5664137418678318

stock future higher:
161 0.9414562362861284
97 0.7279659778248622
159 0.6777561127794045
31 0.6339827914530628
155 0.5977247079893834
38 0.5502905543544416
153 0.5147496793930558
65 0.4927182909743877
76 0.4927182909743877
11 0.4561684973096685
```

Note:
1) We allow some minor tolerance in the result. For example, when compute idf=log10(N/df), if you didn't convert the integer type of N to double, it will lead to some tolerance. Because for the integer type, "/" operation will ignore the fractional part.
A sample of query2 (stock rally) could be the following(without considering the double/integer type issue):

```
190, 0.6709367456337463
165, 0.6211667247837058
145, 0.6211667247837058
102, 0.6211667247837058
8, 0.6211667247837058
82, 0.5480417645255817
26, 0.5480417645255817
186, 0.5073880811125991
```

38, 0.5073880811125991
18, 0.14120766614030913

But this minor tolerance will not lead to wrong results, which means the returned docID should be the same with the above ones. (The order of docID doesn't matter if they are associated with same similarity. For the queries that will return more than 10 results, the last one could be different, according to your sort function.)

2) Computing overall weight of query term:

w(term,Q)=(1+logtf)*log(N/df)

-->e.g. For query "europe", (europe appears in 4 documents, totally we have 200 documents)
w(europe,Q)=(1+log1)*log(200/4)

3)Computing normal weight of term in doc:

nw(term,D)=(1+logtf)/||D||

Here we use (1+logtf) instead of (1+logtf)*log(N/df), and so is for computing ||D||
These two formulas can be used to compute similarity.

4) Should use the document-at-a-time algorithm (described in slide7), not term-at-a-time algorithm.