MapReduce for Large Scale Computing Solutions to Think-Pair-Share Activities

1. **MapReduce Exercise 1:**

   Case Analysis or Capitalization Probability: In a collection of text documents, find the percentage capitalization for each letter of the alphabet. That is, (number of occurrences of a letter that are capitalized/ total number of occurrences for that letter) $\times$ 100.

```
file1: The happy Fox
file2: The THe THE
file3: And ANd AND

result:
a: 3/4 * 100 = 75%
t: 4/4 * 100 = 100%
e: 1/4 * 100 = 25%
. . .
```

   Solution

   The main idea is to output a 2-tuple for both a lower case and upper case character that has the same key: the character in uppercase. The uppercase 2-tuple will have a value of "1" where as the lowercase one would have a value of "0". Then all the 2-tuples for a character will go to one reducer that can add the number of characters as well the number of upper case characters to finally output the percentage.

```
map(String key, String value):
// key: document name
// value: document contents (line by line)
   for each character ch in value:
   if (ch is uppercase)
       emitIntermediate(ch, 1)
   else
       emitIntermediate(toUppercase(ch), 0)


reduce(String key, Iterable values):
// key: a character
// values: a list of counts
   int total = 0
   int sum = 0;
   for v in values:
       sum += parseInt(v)
       total += 1
   emit(key, asString(count * 100.0/total))
```

2. **Top-N patents**

- Find the number of citations for each patent in a patent reference data set. The format of the input is

  citing_patent, cited_patent

  Describe a MapReduce algorithm to solve this problem.

```
map(String key, String value):
// key: document name
// value: citing\_patent, cited\_patent (on a line by itself)
1.  parse into two variables: cited\_patent and citing\_patent
2.  emitIntermediate(cited\_patent, 1)


reduce(String key, Iterable values):
// key: cited\_patent
// values: a list of 1's
1.   int sum = 0;
2.   for v in values:
3.      sum += parseInt(v)
4.  emit(key, asString(sum))
```

- Find the top *N* most frequently cited patents. The format of the input is

  citing_patent, cited_patent

  Describe a MapReduce algorithm to solve this problem.
  Hint: This will take two passes.
  Solution.

```
map1(String key, String value):
// key: document name
// value: citing\_patent, cited\_patent (on a line by itself)
1.  parse into two variables: cited\_patent and citing\_patent
2.  emitIntermediate(cited\_patent, 1)


reduce1(String key, Iterable values):
// key: cited\_patent
// values: a list of 1's
1.   int sum = 0;
2.   for v in values:
3.      sum += parseInt(v)
4.   insert (key, sum) into global priority queue of fixed size (N)
5.   // the priority queue is sorted by sum and only has N entries

setup(): //for reduce1
1.  initialize global priority queue for (patent, count) values, sorted by count

cleanup(): //for reduce1
1.  for each (patent, count) pair in the global priority queue:
2.      emit(patent, count)

map2(String key, String value):
// key: document name
// value: cited_patent, count
1.  parse into two variables: cited\_patent and count
2.  emitIntermediate(cited\_patent, count)

reduce2 --> same as reduce1 (including setup and cleanup)
```