Name: _____
USC ID: _____

INF 553 - FALL 2019

QUIZ 1 (10 Points)

1. [0.5 point] In data mining, the discovered patterns and models are required to be:
    a. Useful
    b. Valid
    c. Understandable
    d. Unexpected

All of the these

2. [0.5 point] The Bonferroni's principle reminds us that:
    a. We may not find suitable models due to chance alone
    b. Not all data mining results are valid
    c. None of these
    d. All of these

All of these

3. [1 point] Clustering is a descriptive method while Recommender systems is a predictive method.

4. [0.5 point] Which of the following is most likely the typical size of chunks?
    a. 64KB
    b. 64MB
    c. 32MB
    d. 32KB

64 MB

5. [0.5 point] How to store data persistently if nodes fail?
Ans. Distributed File System

6. [1 point] MapReduce addresses the challenges of cluster computing by storing data redundantly, minimizing data movement and simple programming model.

7. [1 point] Master node is also known as Name Node in Hadoop's HDFS

8. [1 point] Which of the following functions are typically provided by the MapReduce?
    a. Map
    b. Reduce
    c. Group by

d. Shuffle

All of the above

9. [4 points] For each of the following problem describe how you would solve it using Map-Reduce. Just indicate the logic needed using pseudocode. [No need of python code or any other detailed programming language].

a) Fill in the logic for the **word count** using MapReduce and Using a COMBINER.

Map (key, value):
#value is a document containing words to be counted.

key: document name; value: text of the document

```
for each word w in value:
emit (w, 1)
```

Reduce (key, values):

key: a word; value: an iterator over counts

```
result = 0
for each count v in values:
    result += v
emit (key, result)
```

b) Design a MapReduce algorithm that takes a very large file of integers and produces as output all unique integers from the original file that are evenly divisible by 3.

Map (key, valuelist):

```
for v in valuelist:
    if (v % 3) == 0:
        emit (v, 1)
```

Reduce (key, values):

// Eliminate duplicates

```
emit (key, 1)
```