MapReduce

Use pseudocode to write MapReduce functions necessary to solve the problems below. Also, make sure to fill out the correct data types. Some tips:

- The input to each MapReduce job is given by the signature of the **`map()`** function.
- The function **`emit(key k, value v)`** outputs the key-value pair **`(k, v)`**
- You may use the **`for(var in list)`** syntax to iterate through **`Iterables`**, or you can call the **`hasNext()`** and **`next()`** functions
- Data types you may use are: **`int, float, String`**, list of these primitives, and custom data types composed of these primitives

1. Given a set of classes that students have taken, output each student's name & total GPA.

Declare any custom data types here:

```
CourseData:
    int courseID
    float studentGrade  //a number from 0-4
```

map(String student, CourseData value):

```
emit(student, value.studentGrade)
```

reduce(_____String_____ key,
Iterable< ___float____> values):

```
totalPts = 0
totalClasses = 0
for ( grade in values ):
    totalPts += grade
    totalClasses++
emit(key, totalPts / totalClasses)
```

2. Compute the list of mutual friends between each pair of friends in a social network. Each person on the network is identified by a unique **`int`** ID. The **`intersection(list1, list2)`** method returns a list that is the intersection of **`list1`** and **`list2`**.

Declare any custom data types here:

```
FriendPair:
  int friendOne
  int friendTwo
```

map(int personID, list<int> friendIDs):

```
for ( fID in friendIDs ):
    if ( personID < fID ):
        friendPair = ( personID, fID )
    else:
        friendPair = ( fID, personID )
    emit(friendPair, friendIDs)
```

reduce(___FriendPair____ key,
Iterable< ____list<int>____> values):

```
mutualFriends = intersection(values.next(),
                             values.next() )
emit(key, mutualFriends)
```

3. A. Given a set of coins and each coin's owner, compute the number of coins of each denomination that each person has.

Declare any custom data types here:

```
CoinPair:
    String person
    String coinType
```

map(String person, String coinType):

```
coinPair = (person, coinType)
emit(coinPair, 1)
```

reduce(____CoinPair____ key, Iterable< _____int_____> values):

```
total = 0
for ( count in values ):
    total += count
emit(key, total)
```

B. Using the output of the first MapReduce, compute the amount of money each person has. The function `valueOfCoin(String coinType)` returns a float corresponding to the dollar value of the coin.

map(____CoinPair_____ key, _____int_____ value):

```
emit(coinPair.person,
    valueOfCoin(coinPair.coinType) )
```

reduce(____String_____ key, Iterable< _____float_____> values):

```
total = 0
for ( amount in values ):
    total += amount
emit(key, total)
```

Warehouse-Scale Computing

Power Usage Effectiveness (PUE) = (Total Building Power) / (IT Equipment Power)
Total Building Power = IT Equipment + Power supplies + Networking equipment + Cooling equipment

Sources speculate Google has over 1 million servers. Assume each of the 1 million servers draw an average of 200W, and that Google pays an average of 6 cents per kilowatt-hour for datacenter electricity.

a) Estimate Google's annual power bill for its datacenters. Ignore the power cost of networking equipment. Assume 365 days (8760 hours) in a year.

1,000,000 servers x 0.2kW/server x 0.06 dollars/kW-hr x 8760 hrs/yr = $105.12 M/yr

b) Google reduced the PUE of a 50,000 machine datacenter from 1.5 to 1.25 without decreasing the power supplied to the servers. What's the cost savings per year?

50,000 servers * .2k W/server * (1.5 - 1.25) * 0.06 dollars/kW-hr x 8760 hrs/yr = $1.314 M/yr