1. Map Reduce

(a) (10 points) What are the inputs and outputs of the Map function and the Reduce function.

(b) (10 points) Design a map-reduce algorithm to take a very large file of integers and produce as output as a sequence of the following:
   (number, count)
   i.e., the count of the number of distinct integers in the input.

(c) (10 points) Why is it possible to push some of what reducers do to the Map tasks when the Reduce function is associative and commutative (combiner function)? Give an example when the Reduce function is not associative and commutative and the push does not work.

2. Oracle NoSQL

Oracle has two different NoSQL API’s (Application Programmers’ Interfaces).

NoSQL Database Tables Interface
NoSQL Database Key/Value Interface

In tables API one can define tables with columns while in key/value API one can define keys with associated values. In tables interface some of the columns have to be defined as keys. In both tables API and key/value API the database search is based on keys. Value portion of key/value pair is stored using Avro Schema (similar to JASON). An example of JASON/Avro Schema is as follows:

```json
{
  "type": "record",
  "namespace": "com.example",
  "name": "FullName",
  "fields": [
    { "name": "first", "type": "string" },
    { "name": "last", "type": "string" }
  ]
}
```

type: is defined as record here, i.e., there are multiple fields defined below. For Avro this must always be a record.
Namespace and name: identify the schema and needed by the application to identify which
schema it is using.

fields: This is the actual schema definition which is
essentially defining the type of each field
of the values.

Fields have the following format: keyword "name" followed
by the actual name of the field which is for
example "first" here. type of the field "first"
is string. Some times field comes with a
default value

Type can be nested. For example, we can have
"name": "address", "type": "record",
"name": "city", "string" },

Avro schema is defined for Key/Value API but Oracle does provide a mechanism to
convert an Avro schema into a table (schema) for use in Tables API. Each stored value of
key/value is associated with schema. Values are serialized before storing, i.e., each value
is stored without any meta data except a small internal schema identifier. Thus, one
such reference is stored per key/value pair. An important idea here is that serialized
values are stored in a very efficient binary format. Thus, before writing values are
serialized using the schema and after reading the values are deserialized using the Avro
schema (code stored with the value).

(a) In Tables API one can define child tables. What are child tables and what are the
advantages of defining child tables.

(b) Keys are defined as Major keys, minor keys and shard keys. What are the differences
of their functions.

(c) Take the example of Natural Join by mapReduce of page 13 of the lecture note.
Give Avro schema for the input of map and the output of reduce.

(d) A small internal schema identifier is stored with each value of a key/value pair.
Why is an identifier not stored with a group of values with the same Avro schema.

(e) What are the advantages of default value in a field for Avro Schema?