INF 551 – Spring 2017

Homework #4: DyanmoDB & Hadoop

Due: April 14
100 points

1. **[DyanmoDB , 40 points]** Write two Python scripts. The first one "load.py" creates a DynamoDB table and loads the first 1,000 records in ("data" section of) "lax.json" (the same as that in homework #2 and #3). The second one "search.py" performs similar search function as in homework #3.
    - [30 points] **load.py**:
        - Your DynamoDB should use the *record id* (the first column of the record: 1, 2, etc.) as the key.
        - It should store sufficient attributes for each record to answer the question in the *search.py* script below.
        - It should use *batch_writer()* of *boto3* to load the data in batch for efficiency.
        - Execution: *python load.py lax.json*

    - [10 points] **search.py**:
        - It takes two keywords (that specify the search condition) as the input, and outputs the total number of passengers in the records that satisfy the condition.
        - The first keyword is either "Departure" or "Arrival"; the second is a four-digit year, e.g., 2006. There're **only** and **always** these two kinds of keywords.
        - Your script should turn user search into "scan" function of *boto3* and execute it on the above table you created.
        - Execution: *python search.py Arrival 2015*

    Submission: <FirstName>_<LastName>_load.py & <FirstName>_<LastName>_search.py.

2. **[Hadoop MapReduce, 60 points]** Given a template script "Join.java", fill in the missing codes (look for "fill in here" in the template).
    - Join algorithm will join two input files located in two different directories*: input-age* and *input-weight*. Each directory contains a text file. For example, *ages.txt* under *input-age* directory may have the following content:

    ```
    david    35
    mary     25
    john     53
    jennifer         38
    ```

    Each line contains a person name and his/her age, separated by a tab.

Similarly*, weights.txt* under *input-weight* directory may have the following content:

```
david   46
mary    36
john    56
bill    46
```

Each lines contains a person name and his/her weight (lbs), also separated by a tab.

- The two files will be joined on their first columns (person names). The algorithm produces the joined tuples. That is, it performs a natural (inner join). For example, it produces the following output for the example data above:

```
david   (35, 46)
john    (53, 56)
mary    (25, 36)
```

Each line consists of a person name and a tuple of his/her age and weight.

Submission: the completed <FirstName>_<LastName>_Join.java.