**Readme**

**Synopsis:** The aim of this project was to design a simple distributed key-value storage based on Chord. Although the design is based on Chord, it is a simplified version of Chord; I implemented three things

1) ID space partitioning/re-partitioning,

2) Ring-based routing, and

3) Node joins.

**Implementation:** I used an Android activity and Content Provider to implement the Chord style key value storage. This was tested on five different Android emulators and was checked under scenarios like node failure, node joining and node based dynamic re-routing.

**References:** The algorithm is based on the paper  
 <http://conferences.sigcomm.org/sigcomm/2001/p12-stoica.pdf>

**Running the application:**

1. Install the SimpleDHT.apk and run it from each emulator instance.
2. Your app should have an activity used for testing. It has two buttons, one button that displays “Test” and the other button that displays “Dump.” Here are the requirements for the buttons.
3. Test
   1. When touched, this button should insert ten <key, value> pairs into your DHT and retrieve them by using your content provider’s insert() and query().
   2. Each of the ten insert operations should be preceded by a 1-second delay. This is to prevent potentially concurrent inserts.
   3. After generating all insert operations, all <key, value> pairs just inserted should be retrieved. Again, each of the ten queries should be preceded by a 1-second delay.
   4. Each retrieval should display the <key, value> pair retrieved on the screen. The format of the display is “<key, value>” as a string.
   5. The format of the <key, value> pairs is the following:
   6. Key: the sequence number represented as a string starting from 0 (e.g., “0”, “1”, “2”, etc.)
   7. Value: string “Test” concatenated by each sequence number (e.g., “Test0”, “Test1”, “Test2”, etc.)
   8. Each touch of the button resets the sequence number to 0.
   9. Thus, the sequence of operations should be the following:
      1. 1-second delay
      2. Insert <”0”, “Test0”>
      3. sequence number and repeat the above insertion with the 1-second delay until <”9”, “Test9”>.
      4. 1-second delay
      5. Query <”0”, “Test0”>
      6. Display “<0, Test0>”
      7. Increment the sequence number and repeat the query & display with the 1-second delay until <”9”, “Test9”>.
4. Dump
   1. When touched, this button should dump all the <key, value> pairs stored in your local storage. Since you need to implement a distributed key-value storage based on Chord, each instance stores <key, value> pairs that belong to one partition in the Chord ring. This button should display all local <key, value> pairs on the screen.
   2. The order of <key, value> pairs you display does not matter as long as it shows all locally stored <key, value> pairs.