1. public StringTable(int nBuckets);

- Parameters:
  - nBucket: Set the ‘nBuckets’ parameter to the size the caller wants to create. And it will create array for buckets according to the size of parameter and initialize whole LinkedList array by allocating LinkedList<Record>.
- Return value:
  - Void. There is no return value.

2. public boolean insert(Record r);

- Parameters:
  - r: Set the ‘r’ parameter that the caller wants to add. This function receives ‘Record’ as a parameter and adds it into the StringTable. This function will call the add() function of LinkedList to push the parameter ‘r’.
- Return value:
  - ‘True’ or ‘False’. ‘True’ when the insertion was successful and ‘False’ when a record with the same key is already present in the table.

3. public Record find(String key);

- Parameters:
  - key: Set the ‘key’ parameter that the caller wants to find. This function will find the matching ‘Record instance’ in the bucket of the StringTable with the input parameter ‘key’.
- Return value:
  - ‘Record instance’ or ‘null’. It will return the ‘Record instance’ matching input parameter, or null if it does not exist.

4. public void remove(String key)

- Parameters:
  - key: Set the 'key' parameter that the caller wants to remove. It will find the matching record in the bucket with the given ‘key’ and it will remove the record if it exists.
- Return value:
  - Void. There is no return value.

5. private int toIndex(int hashcode);

- Parameters:
  - hashcode: This function will convert input parameter ‘hashcode’ into a table index. Inside the function, it uses ‘Multipli multiplicative hashing strategy’ to convert a hashcode to a index. As a result the caller can find the appropriate index of the LinkedList array.
- Return value:
  - It will return a int variable. It is the bucket index which is generated by input ‘hashcode’.

6. What is the difference in meaning between the (public) size and the (private) nBuckets data members?

- 'size' means how many Record items are added in the StringTable. 'nBuckets' means the size of LinkedList array.

7. Which method(s) must you call, and in what order, to convert a String to an index into the array of hash buckets?

- Call 'stringToHashcode' first and then call 'toIndex'. 'stringToHashCode' generates hash value from the input parameter and 'toIndex' function will find the proper index for the bucket array.
- `int idx = toIndex(stringToHashCode(key));`

8. Where (i.e. in which method) will you allocate the linked list for each hash bucket? Note that "allocating" an object refers to reserving memory for the object and is usually done with the 'new' keyword in Java, as distinct from simply declaring an object.

- I implemented it on the StringTable constuctor. I think it is proper allocating linked list when creating StringTable object. It may be appropriate that everything is set up before adding items on the StringTable.

9. Your hash calculations will involve the modulo operator '%'. Hashcodes in Java are signed integers; what does the Java modulo operator return when given a negative number? How might this impact your toIndex() function?

- what does the Java modulo operator return when given a negative number?
  - Java modulo operator returns a negative value given a negative number.
- How might this impact your toIndex() function?
  - I should convert negative value into positive value because it is used for the index for the bucket array. So, I add the size of 'nBuckets' to the indices and apply one more modulo operation in the 'toIndex()' fuction. It's like shifting to the right direction! As a result, my toIndex() function always returns a positive index whenever it is called.