Maps and Hash Tables:

Say we have a very small hash table that contains the ID numbers of TAs, mapped to their name.

```
Table hashTable = new Table(5);
hashTable.insert(31,"Juvia");
hashTable.insert(13,"Steve");
hashTable.insert(100,"Will");
hashTable.insert(75,"Gareth");
hashTable.insert(28,"Lizzie");
```

Draw what the hash table will look like after the above code is executed, assuming we use a hash function that takes the key mod the length of the array. Edit: assume linear probing to handle collisions.

Fill in the code on the next page (non-generic hash table with integer keys and String values).

```
public class Table {
    private TableRow[] rows;

    public Table(int tableSize) {
        rows = new TableRow[tableSize];
    }

    // TODO 1: return the default position (index) where this key is stored
    private int hash(int key) {


    }

    // TODO 2: locates the position (index) where the specified key can be found,
    // or where it should be inserted if it is not already in the table
    private int locate(int key) {




    }

    // TODO 3: put the specified value in the table under the specified key
    public void insert(int key, String value) {




    }

    // TODO 4: retrieve the value associated with the given key
    public String lookup(int key) {



    }

    private class TableRow {
        int key;
        String value;
        TableRow(int key, String value) {
            this.key = key;
            this.value = value;
        }
    }
}
```