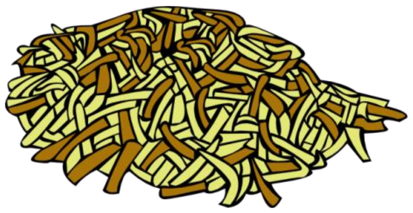
Hash Table Lab

"What I cannot create, I do not understand."



After finishing each part of the lab, copy your entire project and work on the copy for the next part!

Part 1: Implement a simple *HashTable* class.

- All methods take & return *Object* types, but for this lab, you will store < *Integer*, *String*> objects.
- Implement a simple *HashTable* class:

```
public class HashTable
----------------------------------------------------------------------------
         HashTable()               // Sets default table size to 101
         HashTable(int initCap)
 Object  put(Object key,           // Returns the previous value associated with key,
             Object value)         //     or null if there was no mapping for key

 Object  get(Object key)           // Returns the value to which the specified key is mapped,
                                   //     or null if this map contains no mapping for the key

 String  toString()                // Returns a formatted string, ordered by bucket index
```

  - Assume the *initCap* parameter is prime
  - For *put* & *get*, assume there are no collisions.
  - For the *put* method, use the input parameters to build an *Entry* object
  - For the *get* method, unwrap the *Entry* object & return the value
  - When determining the hash index, call the *hashCode* method on the key (external call), then mod with the table size to find the array index
  - For *toString*, make sure to order <*key*, *value*> pairs by array index.
- Implement a simple *Entry* class, as a private inner class of *HashTable*, with public fields:

```
private class Entry
----------------------------------------------------------------------------
         Entry()                   // set key & value to null
         Entry(Object key,
               Object value)

 String  toString()                // return a formatted string for the key & value
```

- Write a driver routine (*main* method) to:
    - Create a *HashTable* object
    - Read a text file (`hash01.txt`) containing < *Integer*, *String*> item pairs
    - The 1st line specifies the starting capacity for the array.
    - The 2nd line specifies the hash table operation & number of operations.
    - Part 1 will only have *put* operations.
    - Save them to the table.
    - Notice the STOP command at the end of the file (not needed in Part 1).
    - Implement a *toString* method returning the saved objects, ordered by bucket index
    - Print the resulting table.
- Test your program by running the *main* method on a small table.
    - Use only non-colliding keys & valid search keys
    - Calculate by hand to validate.
- Turn in using the auto judge

Sample Input (hash1.txt)

```
CAPACITY 17
PUT 12
92800393 LINNIE GILMAN
86770985 DUSTY CONFER
31850991 WANETA DEWEES
46531276 BRADLY BOMBACI
25428367 DUSTY BANNON
68682774 MALIK TULLER
20316453 TOMASA POWANDA
98698743 MALIA HOGSTRUM
81528001 NEAL HOLSTEGE
24248685 FRANCE COELLO
79430806 MELVINA CORNEJO
39977566 CHONG MCOWEN
GET 4
92800393
39977566
46531276
46531277
STOP
```

Sample Output

```
LINNIE GILMAN
CHONG MCOWEN
BRADLY BOMBACI
null
003 : 68682774 MALIK TULLER
004 : 24248685 FRANCE COELLO
005 : 25428367 DUSTY BANNON
006 : 79430806 MELVINA CORNEJO
007 : 98698743 MALIA HOGSTRUM
008 : 20316453 TOMASA POWANDA
009 : 39977566 CHONG MCOWEN
010 : 86770985 DUSTY CONFER
011 : 92800393 LINNIE GILMAN
012 : 31850991 WANETA DEWEES
013 : 81528001 NEAL HOLSTEGE
015 : 46531276 BRADLY BOMBACI
```