Parallel Arrays Programming Challenge

A mobile phone uses the two parallel arrays names and numbers to store up to 100 entries in its

address book. The arrays are globally declared as:

```
public static String[] contactNames = new String[100];
public static long[] phoneNumbers = new long[100];
```

e.g. the first entry might be stored as:

```
contactNames[0] = "sara";
phoneNumbers[0] = 68594753;
```

If the address book is not full, all empty positions are placed at the end of the arrays.

1. Construct the method `static void addEntry(String name, long number)` that adds a new entry (name and number) to the address book, if it is not yet full. Assume that the global variable `entries` stores the current number of entries in the address book (array population).

When the mobile phone receives an incoming phone call, the method `static String findName(long number)` is called, which searches through the address book looking for the telephone number of the caller.

If it is found, the method returns the caller's name. A suitable message is returned if the number is not found.

2. Construct the method `static String findName(long number).`

3. Construct the method `static void sortByName()` to sort the arrays by contact name. Make sure that their respective phone numbers match their respective contacts after the sorting process.

4. Construct the method `static void sortByNumber()` to sort the arrays by phone number. Make sure that their respective contact names match their respective numbers after the sorting process.

5. Construct the method `static String superFindName(long number)` to improve on the `findName(…)` method using binary search. You may want to write a method to check whether the `phoneNumbers` array is sorted (in ascending order) or not; if not, call the `sortByNumber(…)` method before binary-searching for the name.

You will find the expected output on next page →

```
--------------EXPECTED OUTPUT--------------
        Populating arrays...
        Error-database full

        Original arrays
        0: Mick      : 19672022
        1: Peter     : 19671970
        2: Jeremy    : 19671971
        3: John      : 19972022
        4: Christine : 19702022
        5: Dave      : 19721973
        6: Lindsey   : 19742018
        7: Stevie    : 19742922
        8: Bekka     : 19931995
        9: Neil      : 20182022

        Arrays sorted by name
        0: Bekka     : 19931995
        1: Christine : 19702022
        2: Dave      : 19721973
        3: Jeremy    : 19671971
        4: John      : 19972022
        5: Lindsey   : 19742018
        6: Mick      : 19672022
        7: Neil      : 20182022
        8: Peter     : 19671970
        9: Stevie    : 19742922

        Testing findName(19672022) [Mick] : Mick
        Testing findName(19931995) [Bekka] : Bekka
        Testing findName(19702022) [Christine] : Christine
        Testing findName(123456789) [New number] : New number

        Testing superFindName(19972022) [John] : John
        Testing superFindName(19742922) [Stevie] : Stevie
        Testing superFindName(19742018) [Lindsey] : Lindsey
        Testing superFindName(987654321) [New number] : New number

        Arrays after using superFindName (sorted by number)
        0: Peter     : 19671970
        1: Jeremy    : 19671971
        2: Mick      : 19672022
        3: Christine : 19702022
        4: Dave      : 19721973
        5: Lindsey   : 19742018
        6: Stevie    : 19742922
        7: Bekka     : 19931995
        8: John      : 19972022
        9: Neil      : 20182022
```