Bucket Sort and Hash Table with Chaining

Description In this assignment you are requested to implement the bucket sort algorithm and a hash table that handles collisions with chaining, both of which are based on essentially the same data structure. You have to use linked lists for both cases, either from the Standard Template Library (STL) (strongly recommended) or by implementing your own. For usage of STL, refer to `http://www.cppreference.com/wiki`).

1. **Bucket sort**. You can assume that the input will be a set of floats in $[0, 1)$. Testcase number to use with `grade_me`: 41.

2. **Hash table with chaining**. You need to implement insert, search and delete operations. The keys are integers (C++ int type) and you can assume that all keys will be distinct and positive or zero. The first number in the input will be the size m of the chained hash table. Then, each integer of the input will be a key k that you should insert into the hash table. Use the hash function $h(k) = k \mod m$. For collisions, insert the colliding key at the beginning of the linked list. This part of the input terminates when the number -1 is read (such a number must not be inserted in the hash table). After -1 is read, print the content of the hash table to the screen (see the example below for the printing format). Then, read integers from the input until the number -2 is found (do not process that number). For each number inputted, print the pair "i, j", where i is the hash table index and j is the linked list index. If the number is not inside the hash table, print `NOT_FOUND`. Finally, read integers from the input until the number -3 is found (do not process that number). For each number inputted, delete it from the hash table (note that the number might not be in the table). Once the number -3 is found, print the hash table. Testcase number to use with `grade_me`: 45.

3. **Bonus question**. If you did not submit in previous labs the runtime results for the sorting algorithms, you can still do it today.

Input for Test Case 41 The input may contain multiple cases. Each case starts with an integer number which indicates the number of elements to be sorted. Then, the elements follow, one per line. A sequence of 0 elements indicates the end of the input, and produces no output. You can assume the input is correctly structured (i.e. no data is missing).

Output for Test Case 41 Output the sorted sequence one element per line. Do not insert spaces at the beginning or at the end of any element.

Examples of input and output

Testcase 41: input

```
8
0.1
0.9
0.25
0.367
0.478
0.784
0.123
0.483
0
```

Testcase 41: output

```
0.1
0.123
0.25
0.367
0.478
0.483
0.784
0.9
```

Testcase 45: input

```
3
0
3
9
2
6
15
12
-1
```

```
0
2
9
6
5
1
-2
3
15
6
9
1
-3
```

Testcase 45: output

```
12->15->6->9->3->0->

2->
0,5
2,0
0,3
0,2
NOT_FOUND
NOT_FOUND
12->0->

2->
```

Your solutions Before leaving the lab, submit a zipped tar archive of your program through the assignments page of UCMCROPS. Please use your UCMNetID as the filename for the zipped tar archive. Be careful since UCMCROPS strictly enforces the assignment deadlines (deadlines will be every lab date at either 4:20pm or 7:20pm depending on your lab session.).