

---

## Hash Table with Chaining

In this assignment you are requested to implement a hash table that handles collisions with chaining. You have to use linked lists, either from the Standard Template Library (STL) (recommended) or by implementing your own. For usage of STL, refer for instance to <http://www.cppreference.com/wiki>. A sample class structure, with empty methods, is given in the support code. You can either use the given class structure or create your own. In this assignment the keys are integers. Your execution file name must be “Hash.exe”. You will be running “./Grade05” to test your code. Refer to the previous lab assignments for more details on how to use the testing tools.

You have to implement the insert, search and delete operations. The keys are integers (C++ int type) and you can assume that all keys  $k$  are non-negative. The first number in the input will be the size  $m$  of the chained hash table. You will use the simple hash function  $h(k) = k \bmod m$ .

The input contains one of the following commands on a line:

- i key: Insert key into the hash table. For example, “i 2” means “Insert key 2 into the hash table”. For collisions, insert the colliding key at the beginning of the linked list. You just need to insert the key and do not have to output anything in this case.
- d key: Delete key from the hash table. For example, “d 2” means “Delete key 2 from the hash table”. Do nothing if the hash table does not contain the key. If there are multiple elements with the same key value, delete just one element. If the delete is successful, you have to output:

key : DELETED

If not (since there was no element with the given key), output:

key : DELETE FAILED

- s key: Search key in the hash table. If there is an element with the key value, then you have to output:

key : FOUND AT i,j

where  $i$  is the hash table index and  $j$  is the linked list index. If there are multiple elements with the same key value, choose the first one appearing in the linked list. If you do not find the key, then output:

key : NOT FOUND

- o: Output the hash table. Output the entire hash table. Each line should begin with the slot/hash table index followed by key values in the linked list. For example, if  $m = 3$  and we inserted 3, 6, and 1 into an empty table in this order, then you should output:

0 : 6->3

1 : 1

2 :

- e: Finish your program.

## Examples of input and output

### *Input*

```
2
i 4
i 2
i 6
i 3
o
s 2
s 7
d 5
d 2
o
e
```

### *Output*

```
0 : 6->2->4
1 : 3
-
2 : FOUND AT 0,1
-
7 : NOT FOUND
-
5 : DELETE FAILED
-
2 : DELETED
-
0 : 6->4
1 : 3
-
```

**Submission** Submit the source code (only `Hash.cpp` file) through the assignments page of CatCourses by the deadline. Use your UCMNetID as the file name. Be careful since CatCourse strictly enforces the assignment deadline. You do not have to touch the other files. All you have to implement is limited to `Hash.cpp`. You may be asked to compile, run, and explain the code to the TA to prove that you understand what you wrote.

**Grading** We provide 10 test cases for you to try your implementation. Each of them is valued at 1 point if executed correctly. We will use 10 additional test cases to check that your implementation is general enough. These are not provided to you and are also valued at 1 point each.

**Important Note** We will use plagiarism software to detect cheating. Offenders will be subjected to the UCM Academic Honesty Policy which states: *if any violation of the UCM Academic Honesty Policy is suspected in a course, the instructor of record must fill out the Faculty Report for Academic Misconduct.*