

# Implementation of Hash table with linear probing

## 1 Task Description

In this assignment, you have the option to choose C++, Python, or Java for your implementation. Your task is to implement:

- Hash table with linear probing

## 2 Submission Guideline

**You must follow this guideline! Your submission will be marked automatically. Failure to follow this guideline will result in 0.**

Your submission should contain exactly one file, which should be named according to the programming language you choose: `main.cpp` for C++, `main.py` for Python, or `main.java` for Java.

You do not need to submit a design.

You are asked to implement a very specific hash table. The keys are lower-case English words (e.g., apple, pear). The length of a key is at most 10. The hash function is “simply using the last character”. That is, the hash value of apple should be e, and the hash value of pear should be r. Your hash table contains exactly 26 slots (hash value a to hash value z). The total number of English words/keys you need to deal with is at most 26, so the table is never too small.

A table slot has three different statuses: “never used”, “tombstone”, and “occupied”. Table starts with 26 “never used” slots.

**Searching** works as follows: given a key, take its last character as the hash value. First try the corresponding table slot, if the objective key is there, then you have found it. If the corresponding slot is never used, terminate because we are certain that the objective is not in the table. If the slot is occupied but it’s not the objective, or the slot is a “tombstone”, then we move on to the next slot (may need to wrap around the table if the current slot is the last one). We keep trying until we either find the key or are certain that the key does not exist in the table.

**Insertion** works as follows: First perform searching to ensure that the key does not exist. If it already exists, then do nothing. If it does not, take the last character of a key as the hash value. If the corresponding table slot is not occupied (either “never used” or “tombstone”), put the key there (the slot is now occupied). If

the corresponding slot is already occupied, try the next slot. Repeat trying until you find an unoccupied slot.

**Deletion** works as follows: given a key, use the searching process to locate its slot. (If the key is not in the table, then do nothing.) Once you find the key, change the slot status to “tombstone”.

You should start your program by initializing an empty hash table. Your program takes one line as input. The input line contains  $n$  “modification moves” separated by spaces ( $1 \leq n \leq 26$ ). The available modification moves are

- **AWord** (Character A followed by a lower-case English word of length at most 10): **Aapple** means insert key apple into the hash table. If apple is already in the table, do nothing.
- **DWord** (Character D followed by a lower-case English word of length at most 10): **Dapple** means delete key apple from the hash table. If apple is not in the tree, do nothing.

At the end, you need to go through the slots from a to z, and output all the keys separated by space.

You don’t need to worry about invalid inputs.

Sample input 1: Aaaa Accc Abbb

Sample output 1: aaa bbb ccc

Sample input 2: Abba Aaaa Acca

Sample output 2: bba aaa cca

Sample input 3: Abba Aaaa Acca Daaa

Sample output 3: bba cca

### 3 Marking

Marking will be done automatically. The total mark is 10.

- 9 marks for code correctness: your code will be tested against a set of test cases.
- 1 mark for compilation.

### 4 Submission Instructions

You are asked to submit via **Gradescope** <https://www.gradescope.com/> (by either direct upload or via a GitHub repository).

You are welcome to resubmit as many times as you wish before the deadline. We will compile and run your code using the specific commands for each language. The specific compilation commands used by the autograder are:

- For C++: `g++ -std=c++11 -o main.out -O2 -Wall main.cpp`
- For Python: `python3 main.py`
- For Java: `javac main.java` followed by `java -cp . main`

It is your responsibility to ensure that your code compiles and runs correctly on the Gradescope system, as compiler versions and environments may vary (e.g. g++ has too many versions, so being able to compile on your laptop does not guarantee that it compiles on the Gradescope system.)