Name: Sarvagnya H. Purohit **Registration Number**: 201070056
Date: 02-01-2022 **Branch:** Computer Engineering
Course: DSA **Course Instructor:** Dr. Mahesh Shirole

Lab Assignment 7

Aim: Implement Hash Table to store student data using Separate Chaining and Linear Probing. Student data includes RegID (key), name, address, class, grade. Use RegID as key and complete record as value in this implementation.

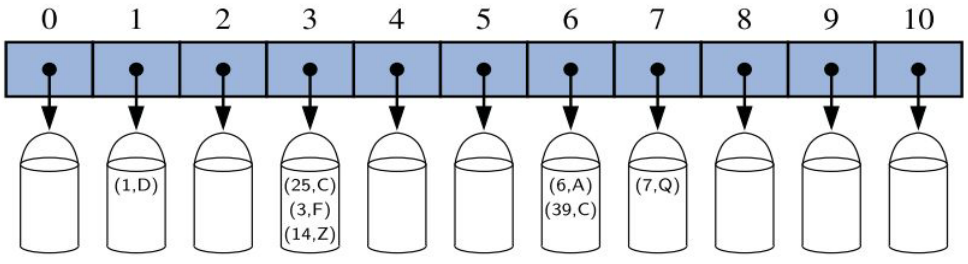
Theory:

Map ADT:
Since a map stores a collection of objects, it should be viewed as a collection of key-value pairs. As an ADT, a map M supports the following methods:

- size(): Returns the number of entries in M.
- isEmpty(): Returns a boolean indicating whether M is empty.
- get(k): Returns the value v associated with key k, if such an entry exists; otherwise returns null.
- put(k, v): If M does not have an entry with key equal to k, then adds entry (k,v) to M and returns null; else, replaces with v the existing value of the entry with key equal to k and returns the old value.
- remove(k): Removes from M the entry with key equal to k, and returns its value; if M has no such entry, then returns null.
- keySet(): Returns an iterable collection containing all the keys stored inM.
- values(): Returns an iterable collection containing all the values of entries stored in M (with repetition if multiple keys map to the same value).
- entrySet(): Returns an iterable collection containing all the key-value entries in M.

Hash Table:
A hash table is a data structure that offers very fast insertion and searching. A map M supports the abstraction of using keys as "addresses" that help locate an entry. A map with n entries uses keys that are known to be integers in a range from 0 to N−1 for some N ≥ n.
If we represent the map using a lookup table of length N, then:

- we store the value associated with key k at index k of the table
- Basic map operations get, put, and remove can be implemented in O(1) worst-case time

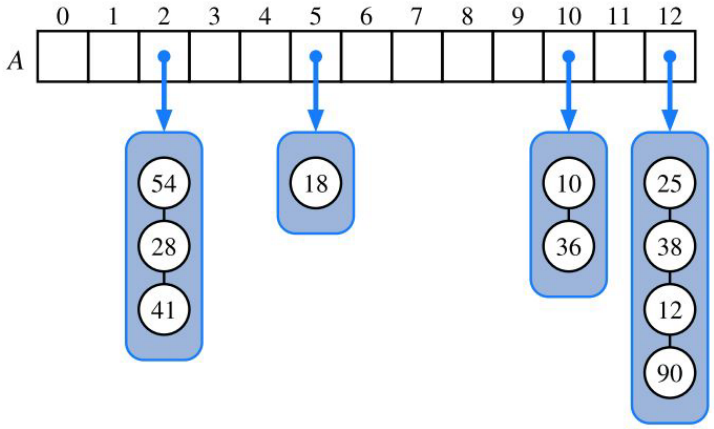


Hash Functions:
The goal of a hash function, h, is to map each key k to an integer in the range [0,N−1], where N is the capacity of the bucket array for a hash table. The main idea of this approach is to use the hash function value, h(k), as an index into our bucket array, A, instead of the key k. That is, we store the entry (k,v) in the bucket A[h(k)]. If there are two or more keys with the same hash value, then two different entries will be mapped to the same bucket in A. This is when we say a "collision" has occurred at that mapped index.

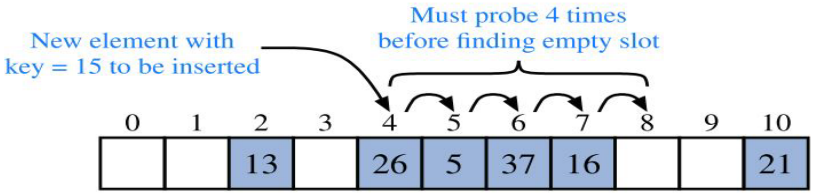
Collision-Handling Methods:
When we have two distinct keys, k1 and k2, such that h(k1) = h(k2) then collisions prevents us from simply inserting a new entry (k,v) directly into the bucket A[h(k)].
There are different ways to address collision:

- Separate Chaining
- Open Addressing

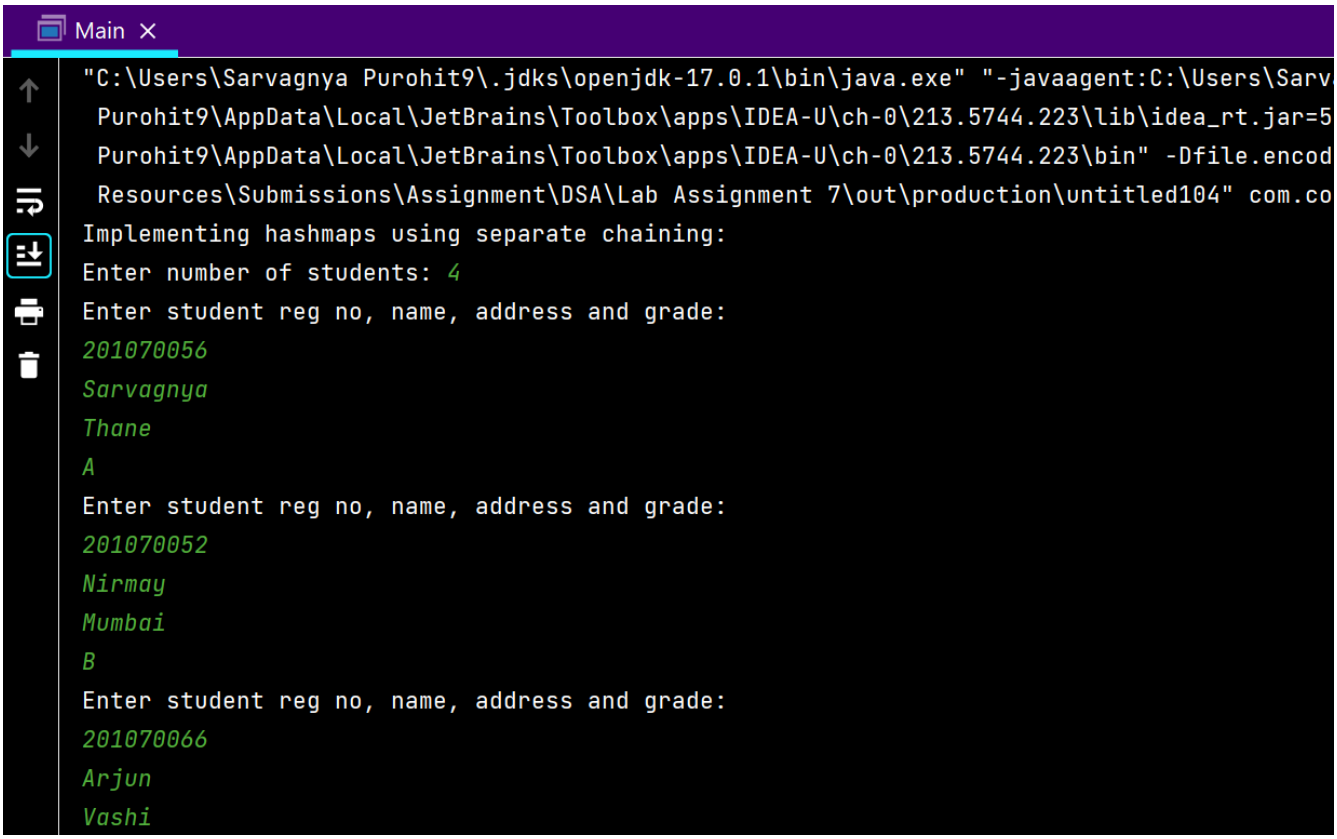
Separate Chaining:
A simple and efficient way for dealing with collisions is to have each bucket A[j] store its own secondary container, holding all entries (k,v) such that h(k) = j.

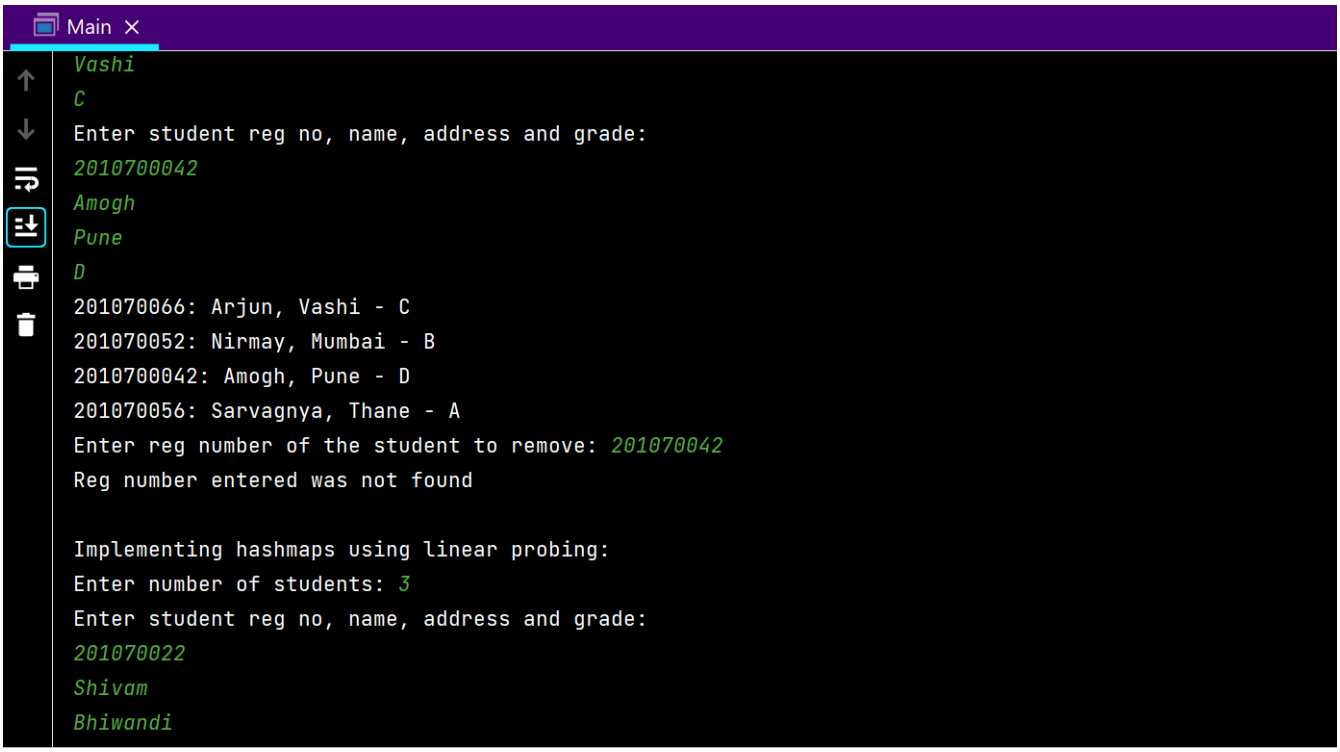


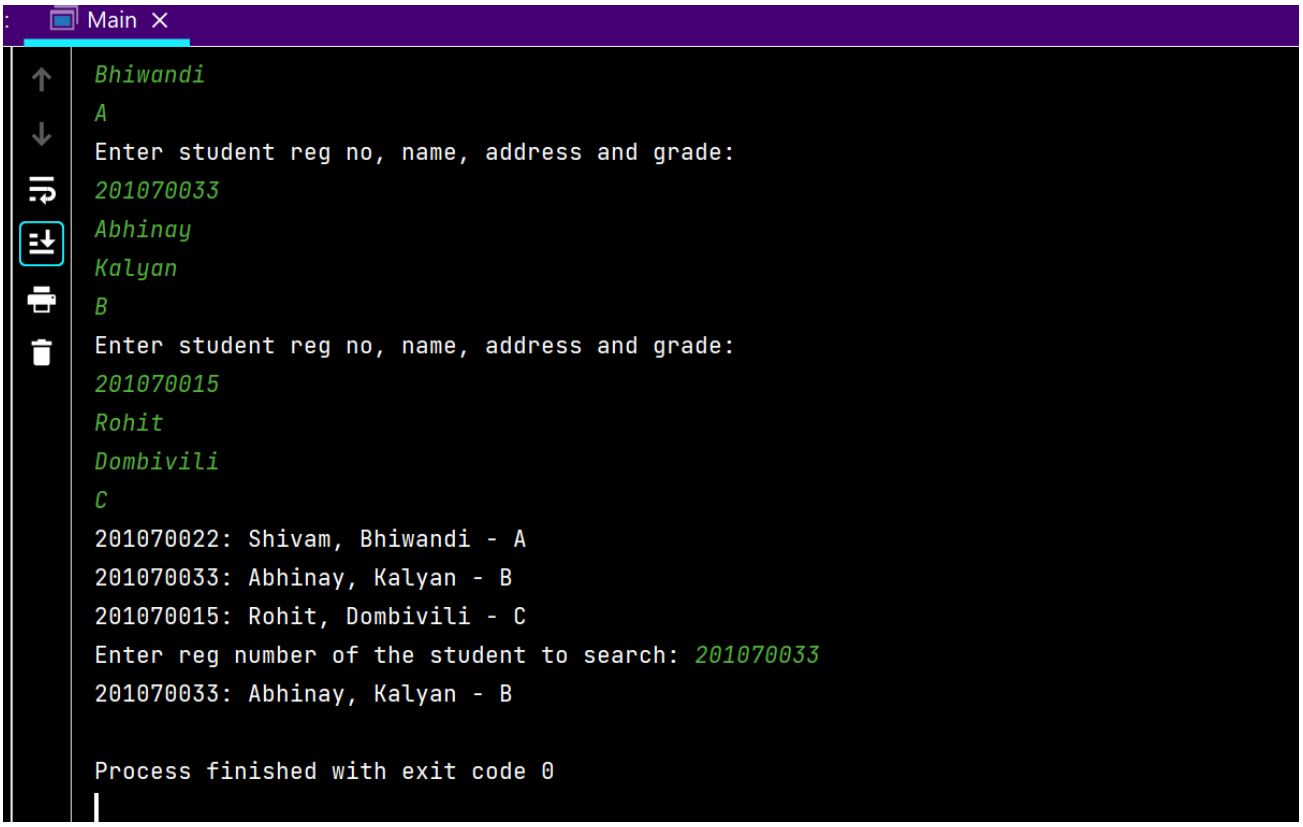
Open Addressing – Linear Probing:
This approach saves space because no auxiliary structures are employed, but it requires a bit more complexity to properly handle collisions.
A simple method for collision handling with open addressing is linear probing. With this approach, ifwe try to insert an entry (k,v) into a bucket A[j] that is already occupied, where j = h(k), then we next try A[(j +1) mod N]. If A[(j +1) mod N] is also occupied, then we try A[(j +2) mod N], and so on, until we find an empty bucket that can accept the new entry. Once this bucket is located, we simply insert the entry there



Test Data and Output:







Conclusion:

Hence, in this lab assignment, we learnt about maps, hashing and the various collision-handling schemes employed in hashing. With hashing, the basic operations like search, insert and delete improve to a O (1) worst case time complexity. We used registration number of students as a key in our hash-map and implemented 2 different methods – linear probing and separate chaining to handle collisions.