# Assignment -1 PS4 (Student Records)

# Technical Design Document - Hash Table

**Purpose**

The purpose of this document is to outline the technical design of the hash table and provides an overview of Implementation and functionality.

* Provides an overview of the technical design of the hash table
* Features considered when designing and Implementing the Hash table

A hash table (hash map) is a data structure that implements an associative array abstract data type, a structure that can map keys to values. A hash table uses a hash method to compute an index into an array of buckets or slots, from which the desired value can be found.

**Implementation Details** Details of defined methods in the class StudentHashTable.

**Method 1:** **initializeHash()**

Initializes an empty list of constant size.

**Input Parameters**

* Self – Current object of hash table

**Method 2: HashId()**

Generates hash value of the Input (studentId) and maps it to a position of bucket list.

First, it evaluates hash value of the input string studentId by taking the sum of Unicode point of each character. Then it uses division method for compression map which basically maps hash value for every student id to a position in the list.

We have used quadratic probing to handle the collisions as it is faster, simple to implement and better at avoiding the clustering problem that can occurs with linear probing.

**Input Parameters**

* Self – Current object of hash table
* studentId – Student Identification Number.
* next\_slot – Initialized with zero and increment if a collision occurs while Insertion.

**Output** – Returns the position for inserting a new record in the bucket list based on the hash value and quadratic probing (In case of collisions).

**Method 3: insertStudentRec()**

Inserts student details in the hash table. It calls HashId() method to get the position of Insertion.

**Input Parameter**

* StudentHashRecord – Current object of hash table
* studentId - Student Identification Number
* CGPA – Student’s CGPA

**Method 4: hallOfFame()**

Returns the list of all students who have a CGPA greater than the CGPA passed as input.

**Input Parameter**

* StudentHashRecord – Current object of hash table
* CGPA – Filter criteria

**Output** – list of all students who have a CGPA greater than the CGPA passed as input.

**Method 5: newCourseList()**

Returns the list of all students who have a CGPA within the given range and have graduated in the last 5 years.

**Input Parameter**

* studenthashrecord – Current object of hash table
* CGPAFrom & CGPATo – Student Score (Fetch from promptPS4 file)

**Output** – list of students based on filter criteria.

**Method 6: depAvg()**

Returns the list of all departments followed by the maximum CGPA and average CGPA of all students in that department.

**Input Parameter**

* StudentHashRecord – Current object of hash table

**Output** – generate txt file (output4.txt) as output, based input criteria

It contains department along with max and average CGPA.

**Method 7: destroyHash()**

Destroys all the entries inside hash table**.**

**Input Parameters**

* StudentHashRecord – Current object of hash table