**4-3 Project One Milestone Two: Hash Table Data Structure**

**unsigned int HashTable::hashFun(int key) {**

**return key % 10**

**}**

**HashTable<String, String> courses = new HashTable<>(8) { (courses table created with 8 buckets)**

Load CSVcourse file

FOR all keys

hash = 0

Use hashCode() FUNCTION to get key’s hash value for a string

((s[0] \* 31 ^ (n – 1)) + (s[1] \* 31 ^ (n – 2)) + … + s(n – 1))

this.hash = key’s hash code value

Make curCor the current node at 0

For all nodes made so far {

IF this.hash > 2000000000 {

curCor.course.cTit = parameter

}

ELSE IF parameter matches a prior node’s key {

IF curCor.course.cPre1 = NULL {

curCor.course.cPre1 = parameter

}

ELSE {

curCor.course.cPre2 = parameter

}

}

ELSE IF parameter does not match any prior node’s key {

curCor.key = parameter

Make a new node called newCor

curCor->newCor

bucket = hashFun(this.hash)

int bucketsProbed = 0

WHILE bucketsProbed < 8 {

IF hashtable[bucket] = NULL {

Hashtable[bucket] = course

}

bucket = (bucket + 1) % 10

++bucketsProbed

}

}

}

}

}

( Keys and their hashCode() values

CSCI100 1773523291

CSCI101 1773523292

CSCI200 1773524252

MATH201 1558971819

CSCI300 1773525213

CSCI301 1773525214

CSCI350 1773525368

CSCI400 1773526174

)

( Organized buckets

0 MATH201

1 CSCI100

2 CSCI101

3 CSCI200

4 CSCI300

5 CSCI301

6 CSCI350

7 CSCI400

)

**displayCourses(Course course)** {

Make curCor the current course at 0

bucketsProbed = 0

WHILE bucketsProbed < 8 {

IF course < 2 parameters {

OUTPUT “Course has less than two parameters!”

}

ELSE {

OUTPUT curCor’s key, title, prerequisite 1 & 2

}

curCor->next

++bucketsProbed

}

}

(Example:

CSCI100 Introduction to Computer Science

CSCI101 Introduction to Programming in C++ CSCI100

CSCI200 Data Structures CSCI101

MATH201 Discrete Mathematics

CSCI300 Introduction to Algorithms CSCI200 MATH201

CSCI301 Advanced Programming in C++ CSCI101

CSCI350 Operating Systems CSCI300

CSCI400 Large Software Development CSCI301 CSCI350

)

**void searchCourse(HashTable<Course> courses, String) {**

int key = hash(course)

int bucketsProbed = 0

Make curCor the current node

IF curCor != nullptr and curCor->course.key = key {

Return curCor->course

}

WHILE curCor != nullptr {

IF curCor->course.key = key {

Return curCor.course.key

}

curCor = curCor->next

}

}