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

+  
  
--------------------------------------------------------------------------------

PSEUDOCODE: ABCU COURSE CATALOG

--------------------------------------------------------------------------------

DATA STRUCTURES & DEFINITIONS

1. STRUCT Course

courseNumber : STRING

name : STRING

prerequisites : LIST<STRING> // zero or more courseNumbers

END STRUCT

2. STRUCT Node

data : POINTER to Course

key : STRING // same as Course.courseNumber

next : POINTER to Node // nullptr if no further chain

END STRUCT

3. CLASS HashTable

buckets : ARRAY of Node\* // array of head pointers, size = tableSize

tableSize : INTEGER

METHOD HashTable(size : INTEGER)

METHOD ~HashTable()

METHOD Insert(coursePtr : POINTER to Course)

METHOD Search(key : STRING) : POINTER to Course

METHOD PrintAll()

END CLASS

4. FUNCTION HashFunction(key : STRING, tableSize : INTEGER) : INTEGER

// Convert key (courseNumber) into an integer rawKey

TRY

rawKey ← CONVERT\_STRING\_TO\_INT(key)

CATCH invalid\_argument

rawKey ← STRING\_HASH(key)

END TRY

RETURN rawKey MOD tableSize

END FUNCTION

--------------------------------------------------------------------------------

MAIN PROGRAM

MAIN

// 1. Read input filename from user

PRINT "Enter course data filename: "

filename ← READ\_LINE()

// 2. Initialize hash table with a chosen size (prime ≥ number of courses)

tableSize ← 11

courseTable ← new HashTable(tableSize)

// 3. Load courses from file into hash table

success ← LoadCourses(filename, courseTable)

IF NOT success THEN

PRINT "Error: invalid file format or missing prerequisites."

EXIT PROGRAM

END IF

PRINT "Courses loaded successfully."

PRINT ""

// 4. Print all courses + prerequisites

courseTable.PrintAll()

// 5. (Optional) Allow user to search for a single course

PRINT ""

PRINT "Enter a course number to find details (or 'quit'): "

inputKey ← READ\_LINE()

WHILE inputKey ≠ "quit" DO

result ← courseTable.Search(inputKey)

IF result == nullptr THEN

PRINT "Course " + inputKey + " not found."

ELSE

PRINT "Found: " + result.courseNumber + " – " + result.name

IF LENGTH(result.prerequisites) == 0 THEN

PRINT "Prerequisites: None"

ELSE

PRINT "Prerequisites: "

FOR EACH p IN result.prerequisites DO

PRINT p + " "

END FOR

END IF

END IF

PRINT ""

PRINT "Enter another course number (or 'quit'): "

inputKey ← READ\_LINE()

END WHILE

// 6. Clean up

DELETE courseTable

PRINT "Good bye."

END MAIN

--------------------------------------------------------------------------------

FUNCTION LoadCourses(filename : STRING, courseTable : POINTER to HashTable) : BOOLEAN

// Returns TRUE if file loaded and validated successfully; FALSE otherwise.

// A. Open file & read all non‐blank lines

fileHandle ← OPEN filename FOR READ

IF fileHandle fails to open THEN

PRINT "Cannot open file: " + filename

RETURN FALSE

END IF

allLines ← EMPTY LIST of STRING

WHILE NOT EOF(fileHandle) DO

line ← READ\_NEXT\_LINE(fileHandle)

trimmed ← TRIM(line)

IF trimmed is NOT EMPTY THEN

APPEND allLines ← trimmed

END IF

END WHILE

CLOSE fileHandle

// B. First pass: split lines into tokens & collect courseNumbers

courseSet ← EMPTY SET of STRING

tokenizedLines ← EMPTY LIST of LIST<STRING>

FOR EACH lineL IN allLines DO

tokens ← SPLIT(lineL, ',') // split by comma

IF LENGTH(tokens) < 2 THEN

PRINT "Format error: each line must have at least courseNumber and name."

RETURN FALSE

END IF

// Trim whitespace from each token

FOR i ← 0 TO LENGTH(tokens) – 1 DO

tokens[i] ← TRIM(tokens[i])

END FOR

courseNum ← tokens[0]

IF courseNum is EMPTY THEN

PRINT "Format error: missing course number on line: " + lineL

RETURN FALSE

END IF

INSERT courseNum INTO courseSet

APPEND tokenizedLines ← tokens

END FOR

// C. Second pass: validate prerequisites, create Course objects, insert into hash table

FOR EACH tokens IN tokenizedLines DO

courseNum ← tokens[0]

courseName ← tokens[1]

// Create a new Course object

coursePtr ← new Course

coursePtr.courseNumber = courseNum

coursePtr.name = courseName

coursePtr.prerequisites = EMPTY LIST

// Handle prerequisites if present

IF LENGTH(tokens) > 2 THEN

FOR i ← 2 TO LENGTH(tokens) – 1 DO

prereq ← tokens[i]

IF prereq is EMPTY THEN

PRINT "Format error: empty prerequisite for " + courseNum

RETURN FALSE

END IF

IF prereq NOT IN courseSet THEN

PRINT "Validation error: prerequisite " + prereq + " for course " + courseNum + " not found."

RETURN FALSE

END IF

APPEND coursePtr.prerequisites ← prereq

END FOR

END IF

// Insert Course object into the hash table

courseTable.Insert(coursePtr)

END FOR

RETURN TRUE

END FUNCTION

--------------------------------------------------------------------------------

METHOD HashTable.HashTable(size : INTEGER)

tableSize ← size

buckets ← ARRAY of size tableSize, each element initialized to nullptr

END METHOD

METHOD HashTable.~HashTable()

FOR i ← 0 TO tableSize – 1 DO

current ← buckets[i]

WHILE current ≠ nullptr DO

toDelete ← current

current ← current.next

DELETE toDelete.data // free Course object

DELETE toDelete // free Node

END WHILE

END FOR

// If buckets array was dynamically allocated, free it here

END METHOD

METHOD HashTable.Insert(coursePtr : POINTER to Course)

// 1. Compute bucket index

bucketIdx ← HashFunction(coursePtr.courseNumber, tableSize)

// 2. Create new Node

newNode ← new Node

newNode.data ← coursePtr

newNode.key ← coursePtr.courseNumber

newNode.next ← nullptr

// 3. If bucket is empty, place newNode as head

IF buckets[bucketIdx] == nullptr THEN

buckets[bucketIdx] = newNode

RETURN

END IF

// 4. Otherwise, find tail of chain and append

current ← buckets[bucketIdx]

WHILE current.next ≠ nullptr DO

current ← current.next

END WHILE

current.next = newNode

END METHOD

METHOD HashTable.Search(key : STRING) : POINTER to Course

bucketIdx ← HashFunction(key, tableSize)

current ← buckets[bucketIdx]

WHILE current ≠ nullptr DO

IF current.data.courseNumber == key THEN

RETURN current.data

END IF

current ← current.next

END WHILE

RETURN nullptr // not found

END METHOD

METHOD HashTable.PrintAll()

FOR i ← 0 TO tableSize – 1 DO

current ← buckets[i]

WHILE current ≠ nullptr DO

PRINT "Course Number: " + current.data.courseNumber

PRINT " Title: " + current.data.name

IF LENGTH(current.data.prerequisites) == 0 THEN

PRINT " Prerequisites: None"

ELSE

PRINT " Prerequisites: "

FOR EACH p IN current.data.prerequisites DO

PRINT p + " "

END FOR

END IF

PRINT "-------------------------------"

current ← current.next

END WHILE

END FOR

END METHOD