CS3743 Program #1 Intro to Hash Files (20 points) Spring 2020

**© Copyright 2020 Larry Clark, this document must not be copied to any other website**

This is the first part of a two-part programming assignment. In this part, we are reading and writing to a hashed file. In the next part, we will be implementing probing to handle synonyms.

I have provided a driver program, include file, and input file. The driver is provided to reduce your effort on this programming assignment.

Files I provided (can be copied from /usr/local/courses/clark/cs3743/2020Sp/Pgm1):

**cs3743p1Driver.c** - driver program which invokes your functions. It also provides a **hash** function.

**cs3743p1.h** - include file which contains constants, HashHeader, HashFile, and Book typedefs, and function prototypes.

**p1Input.txt** - stream input file used by the driver to specify what needs to be invoked

Your hash file will have a header record at Record Block Number (RBN) 0. All other records will either be empty or contain books.

Please see the sample output below.

Functions you must code (each of the functions return either RC\_OK or an error code):

int **hashCreate**(char szFileNm[], HashHeader \*pHashHeader)

This function creates a hash file containing only the HashHeader record.

* If the file already exists, return RC\_FILE\_EXISTS
* Create the binary file by opening it.
* Write the HashHeader record to the file at RBN 0.
* **fclose** the file.
* return RC\_OK.

int **hashOpen**(char szFileNm[], HashFile \*pHashFile)

This function opens an **existing** hash file which must contain a HashHeader record, and sets the pHashFile->pFile.   
It returns the HashHeader record by setting pHashFile->hashHeader..

* Use fopen to open the file. If it doesn't exist, return RC\_FILE\_NOT\_FOUND.
* Use fread to read the HashHeader record and return it through the parameter. If the read fails, return RC\_HEADER\_NOT\_FOUND.
* Do **NOT close** the file since the other functions will assume that it is open.

int **readRec**(HashFile \*pHashFile, int iRBN, void \*pRecord)

This function reads a record at the specified RBN in the specified file.

* Determine the RBA based on iRBN and pHashFile->hashHeader.iRecSize.
* Use fseek to position the file in that location.
* Use fread to read that record and return it through pRecord.
* If the location is not found, return RC\_LOC\_NOT\_FOUND. Otherwise, return RC\_OK.
* Note: if the location is found, that does NOT imply that a book was written to that location. Why?

int **writeRec**(HashFile \*pHashFile, int iRBN, void \*pRecord)

This function writes a record to the specified RBN in the specified file.

* Determine the RBA based on iRBN and pHashFile->hashHeader.iRecSize.
* Use fseek to position the file in that location.
* Use fwrite to write that record to the file.
* If the fwrite fails, return RC\_LOC\_NOT\_WRITTEN. Otherwise, return RC\_OK.

int **insertBook**(HashFile \*pHashFile, Book \*pBook)

This function inserts a book into the specified file.

* Determine the RBN using the driver's hash function.
* Use readRec to read a record at that RBN.
* If that location doesn't exist or the record at that location has a szBookId[0] == '\0':
  + Write the new book record (using pBook) at that location using writeRec.
* If that record exists and that book's szBookId matches pBook->szBookId, return RC\_REC\_EXISTS. (Do not update it.)
* Otherwise, return RC\_SYNONYM. Note that in program #2, we will actually insert synonyms.

int **readBook**(HashFile \*pHashFile, Book \*pBook, int \*piRBN)

This function reads the specified book by its szBookId.

* Since pBook->szBookId was provided, determine the RBN using the driver's hash function.
* Be able to return that RBN via the third parameter. In Pgm#2, the returned RBN might not be the hashed RBN.
* Use readRec to read the record at that RBN. Be careful to not initially overwrite pBook.
* If the book at that location matches the specified pBook->szBookId, return the book via pBook and return RC\_OK.
* Otherwise, return RC\_REC\_NOT\_FOUND

Notes:

1. When looking at your output, your display tool needs to use a fixed font; otherwise with a proportional font, the output will not be aligned properly.
2. The data in your book.dat file will not be easily visible since it contains some binary (e.g., integer) data.
3. Your code must be written based on my **programming standards** and placed in **cs3743p1.c**
4. Do **not** modify either cs3743p1.h or cs3743p1Driver.c.
5. You must run your code on a fox server.
6. Turn in your cs3743p1.c and p1out.txt (output) using a zip file named (*abc123*zip) via BlackBoard. The zip file must not contain any directories.

Sample Output (partial):

\* CS3743 p1Input.txt

\* Nuke the Hash file if it exists

>> NUKE BOOK book.dat

\* Opening of a non-existent file should cause an error

>> OPEN BOOK book.data

\*\*\*\* ERROR: file not found or invalid header record

\*

\* 1. Create the hash file

\*

>> CREATE BOOK book.dat 19 25 5

Record size is 80

>> PRINTALL BOOK book.dat

MaxPrim=19, RecSize=80, MaxOv=25, MaxProbes=5

\*

\* 2. Creating it again should cause an existence error

\*

>> CREATE BOOK book.dat 17 30 5

Record size is 80

\*\*\*\* ERROR: file already exists

\* Open it

>> OPEN BOOK book.dat

\*

\* 3. Insert some books

\*

>> INSERT BOOK JOYPGM001,The Joys of Programming,TECH,PGMING,100

Hash RBN is 8, id='JOYPGM001'

>> PRINTALL BOOK book.dat

MaxPrim=19, RecSize=80, MaxOv=25, MaxProbes=5

8 JOYPGM001 TECH PGMING 100 The Joys of Programming Hash=8

>> INSERT BOOK PYTHON001,Learn Python Without Getting Bit,S PRESS,PGMING,200

Hash RBN is 1, id='PYTHON001'

>> PRINTALL BOOK book.dat

MaxPrim=19, RecSize=80, MaxOv=25, MaxProbes=5

1 PYTHON001 S PRESS PGMING 200 Learn Python Without Getting Bit Hash=1

8 JOYPGM001 TECH PGMING 100 The Joys of Programming Hash=8

>> INSERT BOOK SQLDBB001,Making Your DB Queries SQueeL,XYZ,DB,300

Hash RBN is 16, id='SQLDBB001'

>> INSERT BOOK TECHDR001,My Technical Dream Job,TECH,PGMING,400

Hash RBN is 18, id='TECHDR001'

>> PRINTALL BOOK book.dat

MaxPrim=19, RecSize=80, MaxOv=25, MaxProbes=5

1 PYTHON001 S PRESS PGMING 200 Learn Python Without Getting Bit Hash=1

8 JOYPGM001 TECH PGMING 100 The Joys of Programming Hash=8

16 SQLDBB001 XYZ DB 300 Making Your DB Queries SQueeL Hash=16

18 TECHDR001 TECH PGMING 400 My Technical Dream Job Hash=18

\*

\* 4. Read an existing book

\*

>> READ BOOK TECHDR001

Hash RBN is 18, id='TECHDR001'

18 TECHDR001 TECH PGMING 400 My Technical Dream Job Hash=18

\*

\* 5. Read should not find these two

\*

>> READ BOOK JAVADD001

Hash RBN is 2, id='JAVADD001'

\*\*\*\* ERROR: record not found

>> READ BOOK TECHDR100

Hash RBN is 18, id='TECHDR100'

\*\*\*\* ERROR: record not found

\*

\* 6. Insert two more books

\*

>> INSERT BOOK JAVADD001,Java Isn't an Addiction,S PRESS,PGMING,600

Hash RBN is 2, id='JAVADD001'

>> INSERT BOOK LINUXX004,Learning Linux,XYZ,OS,700

Hash RBN is 10, id='LINUXX004'

>> PRINTALL BOOK book.dat

MaxPrim=19, RecSize=80, MaxOv=25, MaxProbes=5

1 PYTHON001 S PRESS PGMING 200 Learn Python Without Getting Bit Hash=1

2 JAVADD001 S PRESS PGMING 600 Java Isn't an Addiction Hash=2

8 JOYPGM001 TECH PGMING 100 The Joys of Programming Hash=8

10 LINUXX004 XYZ OS 700 Learning Linux Hash=10

16 SQLDBB001 XYZ DB 300 Making Your DB Queries SQueeL Hash=16

18 TECHDR001 TECH PGMING 400 My Technical Dream Job Hash=18

\*

\* 7. Insert an existing book - should cause an error

\*

>> INSERT BOOK TECHDR001,My Technical Dream Job Again,TECH,PGMING,444

Hash RBN is 18, id='TECHDR001'

\*\*\*\* ERROR: record already exists