University of Toronto

**CSCB28F—File Structures, Fall 1998**

# Assignment 2: Quick Searching of Books

*Due: Wednesday 18 November at 10:00 am.*
*Partners: not allowed for this assignment.*
*Worth: 15%.*

## Introduction

The purpose of this assignment is to teach you direct access using various hashing methods. To learn about these techniques, you should read your lecture and tutorial notes on the topic and Chapter 10 of the textbook.

You are to write programs that maintain the book information that you used in assignment 1. You will use the same record structure as the previous assignment. However, in this assignment, you will only have one master and one transaction file, where the only transactions are the add (type = 1), delete (type = 2), and change (type = 3) transactions. All records will again have the *Identifier* field as the key, but the input files do not need to be in sorted order.
The master file records will be taken and stored in a hash file, which is created using the program *hashproca* in part A and *hashprocb* in part B. Once, the hashed master file is created, the transactions from the transaction file will be used to change the hash master file directly. In this assignment, the hash table will have $Numrecords$ records (the number of which is dependent on the part below). You will use the following hashing function to find the initial address: $h(key) = key \bmod Numrecords$.

## Part A: On line processing with linear probing hashing

For this part, you should read the given master file and create a new version of the master file where the position of a record is determined by (linear probing) hashing the key. Then, you have to process the (unordered) transaction file against the hashed master file.
You will use a linear probing hashing scheme where the increment is *Increment* (equal to 7). When a record is deleted, you must signal that the record is free but was occupied before with a special identifier of -2. The identifier value of -1 is used to denote an initially empty space. Finally, do not sort the transaction file; instead, process it sequentially. You will use $Numrecords = 59$.

## Part B: On line processing with separate overflow using chaining

For this part, you should read the given master file and create a new version of the master file where the position of a record is determined by hashing the key using a separate overflow with chaining. Then, you have to process the (unordered) transaction file against the hashed master file.
You will use the overflow with chaining hashing scheme where a collision on a record is handled by adding the new record to a chain in an overflow file pointed to by the record. The pointer will be a 32 bit offset from the start of the overflow file. Every record's overflow chain will be in the same overflow file. When a record is deleted, you must adjust the chain, and also possibly replace the record in the hash file with one in the chain. Deleted records in the overflow will be marked as deleted with the -2 identifier value. Initially empty spaces are marked with -1. You will not reuse deleted spaces in the file. The actual removal of these empty spaces in the overflow would be done by a hash reorganization process, but you will not do this in your assignment. You will use $Numrecords = 23$.

## Your program

For this assignment, your program processes a master file called **master**, and a transaction file called **trans** with the same structure as the master and transaction file as in assignment 1.
In part A, you have to give two programs. The first one should create the **newmastera** file using the hashing scheme for part A. The second one should process the transactions against the **newmastera** file. Call your executable programs **hashmasta** and **hashproca**, respectively. We will use the following commands to run your programs:

```
hashmasta master newmastera
hashproca newmastera trans error.loga
```

In part B, you have to give two programs. The first one should create the **newmasterb** file and the overflow file **overflowb** using the hashing scheme for part B. The second one should process the transactions against the **newmasterb** and **overflowb** file. Call your executable programs **hashmastb** and **hashprocb**, respectively. We will use the following commands to run your programs:

```
hashmastb master newmasterb overflowb
hashprocb newmasterb overflowb trans error.logb
```

Your programs **hashproca** and **hashprocb** must give the average number of records accessed for each kind of operation (insertion, modification and deletion).
Do not read the entire transaction file or the master file all at once into main memory. In real situations, these files may be too large to fit in main memory.

There may be errors in the transaction file and your program should do some error checking. Do not apply the transactions with errors to the master file. Instead, just print to the error file the transaction and a short error message. Your program can consider only errors that depend on the present content of the master file (for instance, a change in a nonexisting book).

## How to hand in your assignment

First, you must submit your files. This involves four programs: `hashmasta.c`, `hashmastb.c`, `hashproca.c`, `hashprocb.c`, and the input and output files. Use the submit command to submit copies of your source codes:

```
     submit -N a2 cscb28f hashmasta.c hashproca.c hashmastb.c hashprocb.c
error.loga error.logb overflowb newmastera newmasterb master trans
```

At the beginning of the class on the deadline, you must hand in source listing for your C programs (Unix) and your data files. This includes your input files (**master** and **trans**), and your output files (**newmastera**, **newmasterb**, and **overflowb** after hashmasta and hashmastb, and **newmastera**, **newmasterb**, and **overflowb** after every 5th transaction in hashproca and hashprocb). When printing out the hash table, you should print the record address beside each newmaster and overflow file record. Skip all blank spaces in the files and do not print them out. For the scheme using overflow with chaining, you should include the pointer address for each record. When printing a record, print the address of the record in the hash and overflow tables, followed by the identifier, the number of copies, and the ISBN. Print out the hash and overflow file at the end of hashmast, and then print the hash and overflow at the end of every 5th transaction.
You do not need to include a report but marks will be given for good style and modularity. The program should be properly documented with features such as comments for each variable and function, variables with sensible names, commented tests in the output files, and so on. The marker must be able to compile your program using **gcc**, and run it successfully on an appropriate test file. You should test your code on other input files as the marker may try different ones herself to test all relevant cases are considered by your code.