UNIVERSITY OF TORONTO
Faculty of Arts and Science

April/May Examinations 1999

CSC228S

Duration — 3 hours

Aids allowed: None.

- **Make sure your examination booklet has 12 pages (including this one).**
- If you run out of space on any question, use the back of a page, and draw an arrow to point this out.
- You will be rewarded for concise, well-thought-out answers, rather than long rambling ones.
- Write legibly. Unreadable answers will be given 0.
- When asked to write code, comments are not necessary.
- If you need to call a standard C function but can't remember the correct order of arguments, just indicate the meaning of each argument.

Family Name: ____________________ Student #: ____________________

Given Names: ____________________

Tutor (circle one): Matthew Cwirko-Godycki Periklis Andritsos Hering Cheng Themistoklis Palpanas Kaytek Przybylski Vincent Corvinelli

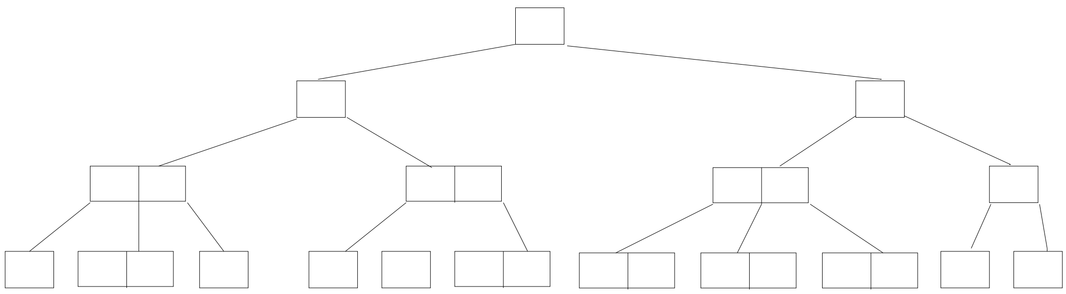
1.	________	/ 11
2.	________	/ 4
3.	________	/ 8
4.	________	/ 12
5.	________	/ 15
6.	________	/ 4
7.	________	/ 10
8.	________	/ 8
9.	________	/ 16
Total	________	/ 88

Question 1

[11 marks in total] Note that this question is about B-trees not B+ trees.

a. [2 marks]
When a record in an internal node is deleted from a B-tree, the deleted node is replaced with another record from the tree. Explain the relationship of the replacement record to the deleted record in terms of their key values.

b. [2 marks]
In the following diagram of a B-tree, the root is going to be deleted. Draw an **X** on the location of the record that will replace the root.



c. [2 marks]
Sometimes the replacement record is in a leaf. Is it always in a leaf? Circle your answer:

YES NO

Explain your reasoning.

d. [5 marks]
In Assignment 2 we used a binary search tree to implement the secondary index on keystring. A B-tree would have been a better choice for this index. Circle **all** of the following that are valid reasons why the B-tree would have been better.

1. The height of a B-tree for the same data is less. So searching requires fewer seeks on average.
2. A B-tree has shorter paths to the more frequently accessed nodes. So searching requires fewer seeks on average.
3. B-tree search is particularly efficient when the tree stores strings.
4. B-trees have more records per node, so after seeking to a node, the payoff is bigger when you read it.
5. Insertion in a B-tree requires writing fewer nodes than insertion in a binary search tree.

Question 2

[4 marks]

Suppose that you have a binary file called `binary.data` which contains with the following bytes:

`00000000 00000000 00000000 00000000 00000000 00000000 0000000 00000000`

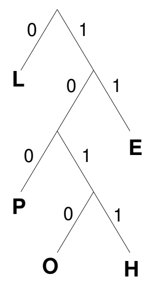
(The above is not output from od — it is just a list of the bytes in the file.) Now suppose you were to run the following program:

```
#include <iostream.h>
#include <fstream.h>

void main()
{
   fstream bin;
   bin.open ("binary.data", ios::in|ios::out);

   int i = 34;
   bin.seekp (2, ios::beg);
   bin.write (&i, sizeof(int));
   bin.close();

}
```

Show exactly what bytes the file would now contain:

Question 3

[8 marks in total]
Assume that the following tree has been built for a file that is to be encoded using Huffman coding.



a. Encode the string "HELLO".

b. The following was produced by encoding a string using the tree above:

10101011110100

Decode this. That is, show the string from which it came:

c. Notice that the tree above is not balanced. In fact, we don't *want* it to be balanced. Explain why not.

d. Outline what information we would need to create the correct tree for encoding a file with Huffman coding.

Question 4

[12 marks in total; 3 marks each]

Below are several design ideas for hashing. For each idea circle the appropriate term to indicate whether it is impossible to implement the idea, it is possible but inefficient, or it is a perfectly fine idea. Then explain why. Do not bother guessing; answers without an explanations will receive no marks.

a. When a home bucket overflows, add room for one more record at the end of the bucket so that the new record will fit.

IMPOSSIBLE POSSIBLE BUT INEFFICIENT A FINE IDEA

EXPLAIN:

b. Use a hash function that has some random component. For example,

$$h(key) = (key \times rand()) \ mod \ F$$

where $rand()$ is some random number generator, and F is a suitable mod factor.

IMPOSSIBLE POSSIBLE BUT INEFFICIENT A FINE IDEA

EXPLAIN:

c. Keep a second file which contains a table of the maximum key in each bucket. Keep this file on disk.

IMPOSSIBLE POSSIBLE BUT INEFFICIENT A FINE IDEA

EXPLAIN:

d. Start the hash table small and use only a few bits of the hashed key to determine the bucket. When a bucket overflows, split it using more of the hashed key to determine the new location of records from the splitting bucket.

IMPOSSIBLE POSSIBLE BUT INEFFICIENT A FINE IDEA

EXPLAIN:

Question 5

[15 marks in total; 3 marks each]

a. Modify the diagram below to make it an inverted file with a secondary index on campus. Assume the secondary index is just a "flat file". Use arrows for any references to the main data file.

Name	Department	Campus
Liszt	Computer Science	Downtown
Kodaly	Physics	Downtown
Byrd	Music	Erindale
Bach	German	Downtown
Palestrina	Computer Science	Scarborough
Victoria	Philosophy	Scarborough
Josquin	Philosophy	Scarborough
Orff	Computer Science	Downtown
Haydn	Computer Science	Downtown
Purcell	German	Scarborough
Handel	Physics	Erindale

b. Modify the diagram below to make it a threaded file with a secondary index on campus. Use arrows for any references to the main data file.

Name	Department	Campus
Liszt	Computer Science	Downtown
Kodaly	Physics	Downtown
Byrd	Music	Erindale
Bach	German	Downtown
Palestrina	Computer Science	Scarborough
Victoria	Philosophy	Scarborough
Josquin	Philosophy	Scarborough
Orff	Computer Science	Downtown
Haydn	Computer Science	Downtown
Purcell	German	Scarborough
Handel	Physics	Erindale

c. For a threaded file, we often store the length of each thread. Explain when this information would be useful.

d. For a threaded file, we don't usually group records into multi-record nodes. Explain why not.

e. In many other file structures, we *do* group records into multi-record nodes or buckets. Explain why we do this.

Question 6

[4 marks in total; 2 marks each]

There are two choices for implementing a "pointer" into a file: One can use (1) a byte offset, or (2) a relative record number.
a. State the advantage of using a relative record number rather than a byte offset.

b. Explain when it is impossible or inappropriate to use a relative record number.

Question 7

[10 marks in total]

The first table below represents a hashed file of records. Each bucket can hold 3 records and the file has a total of 9 buckets. The initial hash function is $h_1(k) = k \bmod 9$ and overflow is handled by double hashing where the step size is determined by a second hash function $h_2(k) = (k \bmod 8) + 1$. The second table represents an array in memory where each array element holds the maximum key value stored in the corresponding bucket.
For each section of this question you must show the resulting file and table once a new record has been inserted. Each part is a separate question. Each time begin again with the **original file.**

a. Insert a record with **30** as the key value by making changes below.

Hashed File

bucket number	keys in bucket		
0			
1			
2	92	65	
3	12		
4			
5	50	14	59
6			
7	7	97	43
8			

Table

bucket number	max key in bucket
0	
1	
2	92
3	12
4	
5	59
6	
7	97
8	

b. Insert a record with **95** as the key value by making changes below.

Hashed File

bucket number	keys in bucket		
0			
1			
2	92	65	
3	12		
4			
5	50	14	59
6			
7	7	97	43
8			

Table

bucket number	max key in bucket
0	
1	
2	92
3	12
4	
5	59
6	
7	97
8	

c. Insert a record with **11** as the key value by making changes below.

Hashed File

bucket number	keys in bucket		
0			
1			
2	92	65	
3	12		
4			
5	50	14	59
6			
7	7	97	43
8			

Table

bucket number	max key in bucket
0	
1	
2	92
3	12
4	
5	59
6	
7	97
8	

d. Insert a record with **34** as the key value by making changes below.

Hashed File

bucket number	keys in bucket		
0			
1			
2	92	65	
3	12		
4			
5	50	14	59
6			
7	7	97	43
8			

Table

bucket number	max key in bucket
0	
1	
2	92
3	12
4	
5	59
6	
7	97
8	

Question 8

[8 marks in total]

Consider a linear hashing scheme, where the initial hash function is $h(k) = k \bmod T$. Recall that when using linear hashing we split a bucket when the performance measure degrades to a certain unacceptable level. Assume we have been adding records to our file and have split a number of times so that at present our file has $T+4$ buckets. Assume T is greater than 10.

a. State the home bucket of a record with key $2T+7$.

b. If we are searching for record with key $3T+1$, state where we should look.

c. We have just inserted record $3T$ and our performance measure degrades to below the acceptable level so that we deem that a split is necessary. State which bucket will split.

d. State the number of the new bucket.

e. For each record in the bucket to be split, explain how we will decide whether to leave it in the original bucket or move it to the new bucket.

f. In order to implement this type of hashing, we need two extra variables that we do not need with ordinary hashing. Outline what two pieces of information they will store.

1.

2.

Question 9

[16 marks in total]

Suppose you are using a relational database to keep track of information for an on-line auction house. The database includes the relations below.

MEMBERS:

memID	Name	email address	Average Monthly Purchase
651123	Fred Flintstone	yabba@flint.com	12.54
669124	Barney Rubble	dabba@flint.com	0

... *etc.*

ITEMS :

ItemID	Description	MemID	minimumBid
102321	cup and saucer	651123	25.00
102322	plate	651123	100.00
102323	roll top desk	665365	650.00
102333	cup and saucer	654234	15.00

... *etc.*

BIDS :

ItemID	memID	Amount
102321	669124	29.99
102321	669126	31.99
102421	669126	51.00

... *etc.*

For this question, you will write a series of queries on this database. **You must use relational algebra.**

a. Write a query to extract the name of member number 669342.

b. Write a query to extract the name of anyone who has bid on any plates.

c. Write a query to extract the email address of the seller of item number 102444.

d. Write a query to extract the email addresses of anyone selling anything on which Joe Smith has placed a bid.

e. Write a query to extract the description of any items up for sale by the member whose email address is “dino@flint.com”.

f. Write a query to extract the ID number of any items bit on by both Barney Rubble and Fred Flintstone.

g. Write a query to extract the minimum bids and itemID for any items for sale by Betty Rubble or Barney Rubble.

END OF EXAM; Total pages = (12)