University of Toronto
csc228S—Introduction to Computing, Spring 1996

Midterm Test

Duration: 50 minutes

Aids allowed: None

Make sure that your examination booklet has 6 pages (including this one).
Write your answers in the spaces provided. You will be rewarded for concise, well-thought-out answers, rather than long rambling ones. Write legibly.

Surname: ______________ First name: ______________

Student #: ______________ Tutor: ______________

1.	______	/ 10
2.	______	/ 4
3.	______	/ 6
4.	______	/ 4
5.	______	/ 6
Bonus[1]	______	/ 1
Total	______	/ 30

[1] Bonus mark for spelling your Tutor's full name properly

Question 1

[10 marks]

Assume that the following code has been executed:

```
typedef struct nodetyp {
   int data;
   struct nodetyp *next;
} Node;
```

Write a C function that takes an unsorted linked list and an integer, and deletes the first occurrence (if there is one) of the integer from the linked list.

```
int delete (Node **headp, int num)

{
    Node            *before;      /* The node before the one to delete */
    Node            *match;       /* The node to delete */

    /* Look through the list to find the node to delete. */
    match = *headp;
    before = NULL;
    while(match != NULL && match->data != num) {
        before = match;
        match = match->next;
    }

    /* If we found a match, now we delete and free it. */
    if(match != NULL) {
        /* Remember the deleted node so that we can later free it. */
        Node *temp;
        temp = match;

        if (match == *headp)
             /* We are deleting the first element.  Reset the head pointer. */
             *headp = (*headp)->next;
        else
             /* We are deleting in the middle.  Point around the element. */
             before->next = match->next;
        free(temp);
    }
    return(0);
}
```

Question 2

[4 marks in total]

(a)

Below is an excerpt from the general master file update algorithm presented in class:

```
if masterRec.key > transRec.key then
      process transRec
      read: transFile, transRec
else ...
```

Considering only the requirements of your program for assignment 2, is it legal for masterRec.key to be greater than transRec.key, or does it represent an error? Circle the appropriate response.

LEGAL $\boxed{\text{ERROR}}$

If it is legal, briefly describe an example transaction for which it is legal; if it is an error, explain why it is an error.

When this condition occurs, it means that *no* master record matches the transaction record's key. This is because no earlier master record has matched, and no later one will either, since the master file is sorted in increasing order by phone number.

In general, it's okay to have a transaction whose key does not occur in the master file. For instance, we might be inserting a new customer. However, there is no insert operation in the program for assignment 2.

(b)

For assignment 2, you will probably use the fwrite function in C to revise file position indicators when you insert new elements into your linked calls file.

Does fwrite write over top of existing bytes in the file, or shift them down to make room for what it is writing? Circle the appropriate answer.

$\boxed{\text{OVERWRITE}}$ SHIFT

Question 3

[6 marks in total]

In this question, we will discuss the merging phase of a sort-merge program, and its efficiency.

(a)

Assume that, in order to merge two sorted files containing a records and b records respectively, a total of $(a+b)$ file writing operations are executed (where each file writing operation writes out one whole record).

Consider 2-way merging, as used in assignment 1. Say you begin with 64 partition files, each containing exactly r records. Over the duration of the merging process, how many times will a file writing operation be executed?

Answer: $\boxed{6 \times 64r}$

Show your rough work here:

See next page

(b)

Now consider 4-way merging, *i.e.*, merging 4 files at once rather than just 2. Assume that, in order to merge four sorted files containing a, b, c and d records respectively, a total of $(a+b+c+d)$ file writing operations are executed.

Again, say you begin with 64 partition files, each containing exactly r records. Over the duration of the merging process, with 4-way merging how many times will a file writing operation be executed?

Answer: $\boxed{3 \times 64r}$

Show your rough work here:

See next page

For 2-way merging:

To go from 64 partitions of size $r \Rightarrow 32$ partitions of size $2r$
32 times, we must merge 2 partitions of size r
$= 32 \times 2r$ writes $= 64r$ writes
To go from 32 partitions of size $2r \Rightarrow 16$ partitions of size $4r$
16 times, we must merge 2 partitions of size $2r$
$= 16 \times 4r$ writes $= 64r$ writes
To go from 16 partitions of size $4r \Rightarrow 8$ partitions of size $8r$
8 times, we must merge 2 partitions of size $4r$
$= 8 \times 8r$ writes $= 64r$ writes
To go from 8 partitions of size $8r \Rightarrow 4$ partitions of size $16r$
4 times, we must merge 2 partitions of size $8r$
$= 4 \times 16r$ writes $= 64r$ writes
To go from 4 partitions of size $16r \Rightarrow 2$ partitions of size $32r$
2 times, we must merge 2 partitions of size $16r$
$= 2 \times 32r$ writes $= 64r$ writes
To go from 2 partitions of size $32r \Rightarrow 1$ partition of size $64r$
1 time, we must merge 2 partitions of size $32r$
$= 1 \times 64r$ writes $= 64r$ writes
IN TOTAL, that is $6 \times 64r = 384r$ writes.

For 4-way merging:

To go from 64 partitions of size $r \Rightarrow 16$ partitions of size $4r$
16 times, we must merge 4 partitions of size r
$= 16 \times 4r$ writes $= 64r$ writes
To go from 16 partitions of size $4r \Rightarrow 4$ partitions of size $16r$
4 times, we must merge 4 partitions of size $4r$
$= 4 \times 16r$ writes $= 64r$ writes
To go from 4 partitions of size $16r \Rightarrow 1$ partition of size $64r$
1 time, we must merge 4 partitions of size $16r$
$= 1 \times 64r$ writes $= 64r$ writes
IN TOTAL, that is $3 \times 64r = 192r$ writes.

In general, the number of writes required is $\boxed{log_w p \times pr}$ where

$w =$ the number of partitions merged at once (in this case 2 or 4)
$p =$ the number of partitions (in this case 64), and
$r =$ the number of records per partition.

Question 4

[4 marks in total]

With some file structures, we *compute* a record's location in a file from its key. There are many methods for doing this. With **direct mapping**, we simply use the key itself as the record's location. With **directory lookup**, we use the key as a location within a directory file. At that location, we store the location of the record itself.

(a)

How does directory lookup save space compared to direct mapping?

With directory lookup, we only make space in the file of records for each actual record, whereas with direct mapping, we must make a space for every *possible* record. That is, one spot per possible key value. If the number of possible records is much greater than the number of actual records, this is a waste of space.

(b)

How does directory lookup *lose* space compared to direct mapping?

With directory lookup, we have to store the directory in addition to the records themselves. That means one pointer for each current record, which isn't stored when we use direct mapping.

Question 5

[6 marks in total]

Imagine that the university is working on a system for storing student information. 9-digit student numbers will be the key field.

(a)

Would direct mapping be a good storage scheme? Circle the appropriate answer.

YES $\boxed{\text{NO}}$

Explain your answer:

```
It would was a great deal of space, because we would need to reserve one spot in the
file for every possible student number, even though only a fraction of them are in
use at any one time.
```

(b)

Assume hashing were to be used, and that we have decided to use 100 buckets. Would a hash function that returns the last 3 digits of the student number be appropriate?

YES $\boxed{\text{NO}}$

Explain your answer:

```
If we were to use the last 3 digits, this would make the range of the hash function
000-999.  So we would need 1,000 buckets, yet we only have 100.
```

(c)

Again assume hashing were to be used, and that we have decided to use 100 buckets. Would a hash function that returns the first 2 digits of the student number be appropriate?

YES $\boxed{\text{NO}}$

Explain your answer:

```
This hash function would have an appropriate range (00-99, hence 100 buckets required),
but it would still be a poor hash function.  The first 2 digits of student numbers
are not evenly distributed among students -- for instance, 94 is very common right
now, but 11 is not.  So there would be a tremendous number of collisions.  An appropriate
hash function would spread out these clusters and thus reduce collisions.
```

END OF EXAM; TOTAL PAGES = 6