University of Toronto
csc228S— File Structures and Data Management, Spring 1998

Midterm Test – Solutions!

Duration: 50 minutes

Aids allowed: Calculator

Family Name: ______________________ Given names: ______________________

Student #: ______________________ Tutor: ______________________

Make sure that your examination booklet has 5 pages (including this one).

For questions that involve writing C 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.

1.	________	/ 16
2.	________	/ 15
3.	________	/ 4
4.	________	/ 4
Bonus[1]	________	/ 1
Total	________	/ 40

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

Question 1

[16 marks in total]

We are writing a program that keeps track of inventory items at a furniture store. For each item, we need to record an item number and the number of units of that item currently in the warehouse.

a. Suppose that we need to create a queue of these items, and that we plan to implement the queue using a linked list. Give a `typedef` for a appropriate C `struct` for nodes in this linked list.

```
typedef struct Node {
      int itemNum;
      int numUnits;
      struct Node *next;
} Node;
```

b. Write a C function called `enqueue` that adds a new inventory item to the back of a queue. Your function should take the following information through its parameters: the item's number, the number of units in the warehouse, and the front and back of the queue (since one or both of these may change). Your function should return 0 if successful, or 1 if something went wrong.

```
int enqueue (int n, int u, Node **front, Node **back) {

    Node *newNode;

    newNode = (Node *) malloc(sizeof(Node));
    if (newNode == NULL)
        return 1;
    else {
        newNode->itemNum = n;
        newNode->numUnits = u;
        newNode->next = NULL;

        if (*front == NULL) {
                *front = (*back = newNode);
        }
        else {
                (*back)->next = newNode;
                *back = newNode;
        }
        return 0;
    }
}
```

c. Assume that we have a queue whose front is pointed to by `front` and whose back is pointed to by `back`. Call your function to add the following new inventory item to the queue: item number 900127, with 46 units in the warehouse. You may ignore your function's return value.

```
enqueue (900127, 46, &front, &back);
```

Question 2

[15 marks in total]
Consider a disk drive with the following properties:

bytes per sector:	1024	average seek time:	5.2 msec
sectors per track:	20	average rotational delay:	8.3 msec
		transfer rate:	2048 bytes/msec

a. Suppose we have a data file containing 5,000 records, each 256 bytes long. How many tracks are needed to store this file? Assume that entire tracks are free, so that we don't have to chop the file into small pieces. (However, these tracks may not be side by side — they could be anywhere on the disk.)

ROUGH WORK:

```
5,000 records * 256 bytes/record     = 1,280,000 bytes to store.

20 sectors/track * 1024 bytes/sector = 20,480 bytes per track.

1,280,000 bytes / 20,480 bytes/track = 62.5 tracks required.

          AN ALTERNATIVE DERIVATION OF THE ANSWER:

1024 bytes/sector / 256 bytes/record = 4 records/sector.

4 records/sector * 20 sectors/track  = 80 records/track.

5,000 records / 80 records/track     = 62.5 tracks.
```

FINAL ANSWER: $\boxed{62.5}$ tracks. (63 also accepted as an answer.)

b. How long would it take to read this file from the disk if we read the file in sequence from beginning to end? Assume that when we get to a track, we can read the whole track without interruption.

ROUGH WORK:

```
1 seek/track * 5.2 msec/seek             = 5.2 msec/track for seeking

1 rotational delay/track * 8.3 msec/r.d. = 8.3 msec/track for r.d.

20,480 bytes/track / 2048 bytes/msec     = 10 msec/track for data transfer

62 * (5.2 msec/track +  8.3 msec/track + 10 msec/track) in total for full tracks
  +
1  * (5.2 msec/track +  8.3 msec/track +  5 msec/track) in total for last track

        = 62 * 23.5 + 18.5 msec
        = 1457 + 18.5 msec
        = 1475.5 msec

The key point here is that the seek time and rotational delay are per *track*.
```

FINAL ANSWER: $\boxed{1475.5}$ msec.
(If presume all of the last track must be read, 1480.5 msec also accepted.)

c. Suppose that in addition to the data file, which is not sorted, we also have an index on that file. The index is stored as an array in memory and *is* sorted according to some key. How much time would be spent reading the file if we were to read it in order by key, using the index to tell us where to find the records in order? Count only time spent reading the file; ignore all time required to go through the index itself.

ROUGH WORK:

```
1 seek/record * 5.2 msec/seek           = 5.2 msec/record for seeking

1 rotational delay/record * 8.3 msec/r.d. = 8.3 msec/record for r.d.

1024 bytes/sector / 2048 bytes/msec      = .5 msec/sector for data transfer

Therefore, we need .5 msec/record for data transfer, because the smallest amount
we can transfer is a whole sector.

5,000 * (5.2 msec/record +  8.3 msec/record + .5 msec/record) in total

        = 5,000 * 14 msec

        = 70,000 msec

The key point here is that the seek time and rotational delay are per *record*,
leading to a much greater amount of time required.
```

FINAL ANSWER: $\boxed{68{,}125}$ msec.

d. Complete the following declarations, which create the index in memory:

```
#define INDEX_SIZE 5000

typedef struct {
       int          key;       /* Key value of a record */
       __long int__ location;  /* Location of the record in the data file */
} indexEntry;

__indexEntry__ index[INDEX_SIZE];
```

e. Write a single line of C code that writes the entire index to a file. Assume that all 5,000 elements of the index array are in use, and that we already have an appropriate `FILE *` called `indexFile`.

```
int numWritten = fwrite (index, sizeof(indexEntry), INDEX_SIZE, indexFile);
```

Question 3

[4 marks in total; 2 marks each, and a **deduction** of 1 per wrong answer]

a. Say we have an integer `n` stored in a variable in memory, and we plan to write it to a file. Which of the following is true? Circle the one best answer.

1. It will consume **less** space if written in binary format rather than text format.
2. It will consume **more** space if written in binary format rather than text format.
3. It will consume **the same** amount of space if written in binary format rather than text format.
4. We don't have enough information to determine whether it will consume more or less space if written in binary format rather than text format.

b. Say we have a string `s` stored in a variable in memory, and we plan to write it to a file. Which of the following is true? Circle the one best answer.

1. It will consume **less** space if written in binary format rather than text format.
2. It will consume **more** space if written in binary format rather than text format.
3. It will consume **the same** amount of space if written in binary format rather than text format.
4. We don't have enough information to determine whether it will consume more or less space if written in binary format rather than text format.

Question 4

[4 marks in total]

a. Fill in the following table to show constraints on the contents of a node in a B-tree of order 7:

	minimum	maximum
number of data values per node:	3	6
number of pointers per node: (null pointers if the node is a leaf)	4	7

b. Fill in the following table to show the special rule for the root node:

	minimum	maximum
number of data values in the root node:	1	6
number of pointers in the root node: (null pointers if the root is a leaf)	2	7