University of Toronto
csc228S - File Structures and Data Management, Spring 1999

Midterm Test

Friday February 26, 1999

Duration: 50 minutes

Aids allowed: None

Family Name: ____________________ Given names: ____________________

Student #: ____________________ Tutor: ____________________

- There are 6 pages, including this one. The test is out of 35 marks and the value of each question is provided; please use this information to manage your time effectively.
- For questions that involve writing code, comments are not necessary. If you need to call a standard method but can't remember the correct order of arguments, just indicate the meaning of each argument.

Part A:	______	/ 8
Part B:	______	/ 4
Part C:	______	/ 13
Part D:	______	/ 6
Part E:	______	/ 4
Total	______	/ 35

Part A [8 marks in total]

The file `mytree.bin` contains the representation of a binary tree of integer triplets. (That is, each node of the tree holds three integers.) Each 16-byte record in the file represents a node of the tree. The root of the tree is not necessarily the first record in the file.

There are no delimeters within or between records. Within each record, the first 2 bytes hold a pointer to the left child of the node and the last 2 bytes hold a pointer to the right child. Both pointers are the byte offset of the child record. The integer value -1 is used, to represent a null pointer; for the machine we used -1 is represented by the octal value 177777. The remaining 12 bytes in each record hold three 4-byte integers.

Below is the output from running `od` on the file:

```
0000000 000100 000000 000007 000000 000000 000000 000007 000140
0000020 000060 000000 000007 000000 000002 000000 000007 000000
0000040 177777 000000 000007 000000 000003 000000 000007 177777
0000060 177777 000000 000007 000000 000004 000000 000007 177777
0000100 000120 000000 000007 000000 000005 000000 000007 000040
0000120 177777 000000 000007 000000 000006 000000 000007 177777
0000140 177777 000000 000003 000000 000001 000000 000000 177777
0000160
```

Reminder: Each line of this od output contains 7 octal digits to show an offset, followed by information about the sixteen bytes of the file that begin at that offset. The bytes themselves are shown using 3 octal digits, but are clumped into pairs (such as `000100`).

1. Draw a picture of the tree that is stored in the file. Label each node with the triplet of integers contained in the node. [4 marks]

2. Is there a *different* file that uses the same record structure and represents an identical tree? Circle the correct answer. [2 marks; -1 for a wrong answer]

   Yes No

3. It might be a good idea to explicitly store in the file the location of the root. Would it be better to store this at the beginning of the file (before any records) or at the end of the file? Circle the best choice. [2 marks; -1 for a wrong answer]

   beginning end

   Briefly explain your reasoning.

Part B [4 marks in total; 2 marks each; no penalty for wrong answers]

Circle the one best answer for each question.

1. With Huffman coding, we build a tree in which any path from the root to a leaf represents the encoding of the character stored at that leaf. Why don't we also store a character at each internal node, and treat the path to the internal node as that character's encoding?

   (a) Maintaining balance in such a tree would be more costly.
   (b) Search in such a tree would be more costly.
   (c) The branching factor in such a tree would have to be less, so we would have taller trees.
   (d) We couldn't decode using such a tree.
   (e) All of the above.

2. The worst case for Btree insertion occurs when

   (a) There is no room in the appropriate leaf, so we add a new layer of leaves at the bottom.
   (b) There is no room in the appropriate leaf, so we cannot insert the new value.
   (c) All nodes are full, so we cannot insert the new value.
   (d) There is no room in the appropriate leaf, and we split nodes and promote all the way to the root.
   (e) There is no room in the root node, and we split nodes and promote all the way to a leaf.

Part C [13 marks in total]

Here are descriptions for three different run length encoding schemes for compressing files of characters:

- A Encode every run (including runs of length 1) with a run length followed by the character. Use 8 bits to represent the run length.
- B Encode every run (including runs of length 1) with a run length followed by the character. Use an integer to represent the run length.
- C Encode only runs of length $> k$ characters, for some carefully selected integer k. Indicate compressed runs with a special value, followed by an integer run length, followed by the character.

Answer the following questions about these three encoding schemes.

1. For scheme A, give the maximum run length that can be represented with our encoding. [1 mark]

2. For scheme C, we need to determine a value for k which will give optimal compression. Assume that the size of the special value is sizeof(char).

   (a) Give a specific value for k assuming that sizeof(char) is 1 and sizeof(int) is 4. [2 marks]

   (b) Because sizeof(char) and sizeof(int) are machine-dependent constants, give a general formula for k in terms of sizeof(int) and sizeof(char). [2 marks]

3. For each of the scenarios below indicate which of the three schemes would be the best by writing the corresponding letter in the blank. [2 marks each; no penalty for wrong answers]

_______ The uncompressed file is mostly composed of single characters but contains some very long runs of blanks.

_______ The uncompressed file is mostly white space (blanks) but has a very small number of of other single characters.

_______ The uncompressed file is mostly composed of single characters or pairs of characters.

_______ The uncompressed file is mostly composed of runs of 3–10 symbols.

Part D [6 marks in total]

Assume that you have a data file that is already open. Given a relative record number, write a C++ method to go to the beginning of that record in the file in preparation for reading it. Include in the parameter list all information that is needed by the method; do not assume that you have access to any variables outside the method.

Part E [4 marks]

For assignment 2, you are creating a linked list in the URL file. There is a tradeoff to be made when deciding on an appropriate node size.

Using at most two sentences, describe what problem(s) would arise from choosing a very large node size.

Using at most two sentences, describe what problem(s) would arise from choosing a very small node size.

You should not need to write below this line.

End of Examination