L0101 Midterm 2002

CSC 228H
University of Toronto

Duration — 50 minutes

Student Number: |_|_|_|_|_|_|_|_|_|

Last Name: ______________________________

First Name: ______________________________

TA: ______________________________

---

*Do **not** turn this page until you have received the signal to start.*

Midterm aids allowed:

- NONE: No Calculators.

Please write legibly.
If you run out of space on a question, use the back of the page.

# 1:	______/10
# 2:	______/ 5
# 3:	______/10
# 4:	______/ 5
TOTAL:	______/30

Good Luck! PLEASE HAND IN

Question 1. [10 MARKS]

Consider an image file format which has a variable length header. Here is the information for the header:

bytes	Purpose
4	size of file in bytes
4	width of image in pixels
4	height of image in pixels
10	ignore
1	offset from beginning of file to pixel data
??	ignore
remaining	pixel data

Part (a) [4 MARKS]

Below are the first few lines of output from running the od program on the same image files. The first was using od with no arguments and the second was using `od -b`. On **one** of the files, circle the byte which holds the pixel data offset and underline the pixel data. You may choose to put your answer on either version of the od output. If you answer on both files and one is correct but the other is wrong you will not get any marks.

Output from running `od` on file 1:

```
0000000 031111 031111 031111 031111 031111 031111 031111 031111
0000020 031111 031111 031111 031111 031111 031111 031111 031111
0000040 031111 031111 031111 031111 031111 031111 031111 031111
0000060 031111 031111 031111 031111 031111 031111 031111 031111
0000100 031111 031111 031111 031111 031111 031111 031111 031111
0000120 031111 031111 031111 031111 031111 031111 031111 031111
```

Output from running `od -b` on file 1:

```
0000000 062 111 062 111 062 111 062 111 062 111 062 111 062 111 062 111
0000020 062 111 062 111 062 111 062 111 062 111 062 111 062 111 062 111
0000040 062 111 062 111 062 111 062 111 062 111 062 111 062 111 062 111
0000060 062 111 062 111 062 111 062 111 062 111 062 111 062 111 062 111
0000100 062 111 062 111 062 111 062 111 062 111 062 111 062 111 062 111
0000120 062 111 062 111 062 111 062 111 062 111 062 111 062 111 062 111
```

ANSWER:

The 23rd byte holds the offset, and since it os only 1 byte long we don't have to worry about weird integer conversions, so $offset = 062$ (octal). Using that information we know the raw data starts in the 000060 row, 3rd column of `od -b`.

```
0000000 ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ...
0000020 ... ... ... ... ... ... 062 ... ... ... ... ... ... ... ... ...
0000040 ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ...
0000060 ... ... 062 111 062 111 062 111 062 111 062 111 062 111 062 111
0000100 062 111 062 111 062 111 062 111 062 111 062 111 062 111 062 111
0000120 062 111 062 111 062 111 062 111 062 111 062 111 062 111 062 111
```

Part (b) [6 MARKS]

Write `c++` code which will read the first value for the first byte of pixel data for any image of this format.

Question 2. [5 MARKS]

Consider a new operating system which has a block size of 2^{10} bytes and uses 16 bit pointers. The inode design is to have 10 direct pointers, 1 pointer to a 1-level index, 2 pointers to 2-level indexes and 1 pointer to a 3-level index.

Part (a) [1 MARK]

How many pointers will fit in a block? _______________

ANSWER: $2^{10}/2^1 = 2^9$

Part (b) [1 MARK]

How many disk accesses will be required to reach block 600? _______________

ANSWER: The first level index addresses blocks 11 to $2^9 + 11$ and node 600 doesn't fit in that interval. It does fit in the set of blocks indexed by the 2nd level index.
That makes 2 accesses to read the index block, + 1 access to read the file block = 3.

Part (c) [3 MARKS]

Calculate the average number of disk reads to read a random block of a file which is 1000 bytes long.

ANSWER: 1

1000 bytes fit perfectly in a block. Then the file will be pointed only by the first direct pointer, which is in memory already. For every read you do in the file only one read is required.

OR if your TA corrected the question to say 1000 *blocks* then the answer is calculated as follows:

```
  10 blocks are accessed with 1 read  = 10
 512 blocks are accessed with 2 reads = 512* 2 = 1024
```

```
  1000- 512 -10 blocks (all the rest) are accessed in 3 reads = 478 * 3 = 1434

add those accesses and divide by 1000 to get the average gives 2.484 disk accesses per block.
```

Question 3. [10 MARKS]

Consider an image file containing only 8 different colour values for the pixels. The colours are shown in the table below along with the freqencies of each colour. Notice that the colours contain a tuple representing a value between 0 and 255 for each of red, green and blue.

Colour	Frequency
(255,0,0)	0.05
(0,255,0)	0.05
(0,0,255)	0.05
(125,217,4)	0.05
(0,255,255)	0.09
(255,0,255)	0.01
(0,0,0)	0.1
(255,255,255)	0.6

Part (a) [1 MARK]

How many bits per pixel (on average) would it take to represent this image if you stored the full RGB colour value for each pixel in the image?

ANSWER: 24. Each colour can range from 0 to 255, 8 bits per color will suffice. Since we have 3 colors components, 3 times 8 bits will encode a full RGB colour

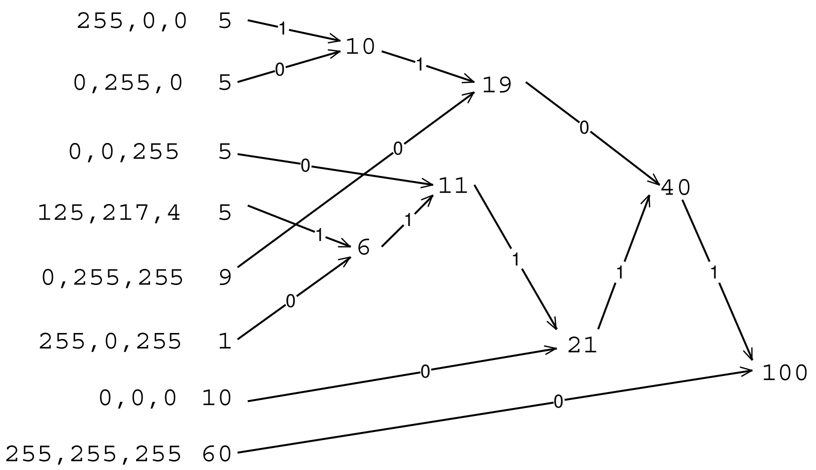
Part (b) [1 MARK]

How many bits per pixel (on average) would it take to represent this image if for each pixel you stored an entry to the colour table using a fixed-length code? You do NOT have to account for the space taken by the colour table in calculating your average. Assume that the image is VERY VERY large and this space in negligible.

ANSWER: 3. Since the table contains 8 colours, it suffices to use 3 bits co represent them ($2^3 = 8$).

Part (c) [2 MARKS]

Show one tree you could build to generate a set of Huffman codes for this file.



Part (d) [2 MARKS]

Give the complete entry that would be stored in the colour table to represent all black including its colour value and its variable length code.

ANSWER: **[0 0 0 011]** (a huffman code of length 3)

Part (e) [4 MARKS]

How many bits per pixel (on average) would it take to represent this image if for each pixel you stored an entry to the colour table using a variable-length code? Again you do NOT have to account for the space taken by the colour table in calculating your average.

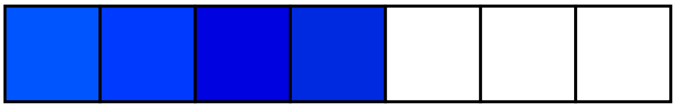
ANSWER: (using the above huffman tree) The average length is

$1/100 * (5 * 4b + 5 * 4b + 5 * 4b + 5 * 5b + 9 * 3b + 1 * 5b + 10 * 3b + 60 * 1b) =$
$1/100 * (20 + 20 + 20 + 25 + 27 + 5 + 30 + 60)b = 207b/100 = 2.07b$

Question 4. [5 MARKS]

Part (a) [1 MARK]

Draw a node for a B-tree with branching factor of 7. Show the minimum number of key/pointer pairs assuming the node is not a root node.



Part (b) [2 MARKS]

What is the maximum height for a B-tree of order 12 which indexes 600 data records?

ANSWER: <u>4</u>
For maximum height we want to minimum number of pointers per node, 2 for root note, 6 for other nodes. That gives us 12 pointers at level 2, 72 at level 3 and 432 at level 4 and ¿600 for level 5. So the maximum height must be 4.

Part (c) [2 MARKS]

What is the minimum height for a B-tree of order 256 which indexes 2^{50} data records?

ANSWER: <u>7</u>
For minimum height we want the tree nodes full. We can get 256 or 2^8 pointers per node. That gives us 2^8 pointers from the root, 2^{16} pointers from level 2, 2^{24} from level 3, ... 2^{48} from level 6, and 2^{56} from level 7. These are the maximum numbers of pointers so 6 is not enough but 7 will do.