University of Toronto
csc228S - File Structures and Data Management, Spring 2000

Midterm Test — Afternoon Section

Friday February 25, 1999

Duration: 50 minutes

Aids allowed: None

Family Name: ______________________ Given names: ______________________

Student #: ______________________

Tutor (circle one): Sirish Pande Kiran Chaudhry Anna Evlogimenou

- There are 6 pages, including this one. The test is out of 30 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:	______	/ 2
Part B:	______	/ 4
Part C:	______	/ 5
Part D:	______	/ 4
Part E:	______	/ 3
Part F:	______	/ 3
Part G:	______	/ 9
Total	______	/ 30

Part A [2 marks in total]

Suppose we have a file of integers, in binary format. The following C++ code attempts to go to the integer that is 9 away from the beginning of the file, and then read and print that integer. But what it always prints is the very first integer in the file. Fix the bug by writing neatly directly on the code. Do not make other improvements to the code.

```
          file.seekp( 9 * sizeof(int),  ios::beg );
          int num;
          file.read( &num, sizeof(int) );
          cout << num;

Answer:   The seekp should be seekg !
```

Part B [4 marks in total]

The following function will do binary search on a file:

```
int BinarySearch(fstream & fs, Record & t){

        Record try;
        int low=0; int high;
        fs.seekg(0,ios::end);
        high=((fs.tellg()-1)/Record::recordSize);
        while(low<high){
                int middle=(high+low)/2;
                fs.seekg(Record::recordSize*middle);
                fs >> try;
                if(try.compare(t)==0)return(try);
                if(try.compare(t)<0)low=middle+1;
                else high=middle-1;
        }
        return(low);
}
```

List four assumptions it makes about class Record.

1. Records are of fixed size in the file.

2. Record has a public int called "recordSize" which is that size.

3. Record has the ">>" operator overloading for reading.

4. Record has a compare function that takes another Record and returns an integer < 0, = 0, or > 0 as usual.

Part C [5 marks in total; **no marks** will be given if your variables are not defined.]

1. State how much time is required in order to merge two sorted files. Give your answer in big-oh terms and define any variables that you use.
   `O(n + m), where n is the number of records in one file, and m is the number of records in the other file.`

2. State how much space is required, in memory, in order to merge two sorted files. Give your answer in big-oh terms and define any variables that you use.
   `O(1).`

Part D [4 marks in total]

Exactly how many disk accesses are required, in the worst case, to read a single block of a unix file if the file contains 2^{13} bytes? Assume that the inode for the file is already in memory, that a block holds 2^{12} bytes, and that a pointer to a file block requires 2^2 bytes. (These are the same numbers we used in class.)

ANSWER: 1 disk accesses

ROUGH WORK:

- The file will fit in two blocks, since it contains 2^{13} bytes and a block holds 2^{12}.
- So only two pointers are needed from the inode to the file's blocks.
- The 12 pointers contained directly within the inode are plenty for this.
- So either of the file's two blocks can be accessed with only one file access – to follow the appropriate pointer from the inode to the file.

I actually meant this question to say that the file contains 2^{13} *blocks*, not bytes. That makes the answer more interesting:

- The first 12 blocks of the file will be accessible directly from the 12 pointers inside the inode.
- The next blocks of the file will be accessible from the 1-level index in the inode.
  - The 1 node in that 1-level index holds 2^{12} bytes, and there are $2^2 = 4$ bytes per pointer, so that node can hold $2^{12}/2^2 = 2^{10}$ pointers to blocks of the file.
- So far we can refer to $12 + 2^{10}$ blocks of the file, but 2^{13} is a little more than that. So the next blocks of the file will be accessible from the 2-level index in the inode.
- That index has a root with 2^{10} pointers, pointing to 2^{10} children that each point to 2^{10} file blocks. So the 2-level index can point to $2^{10} \times 2^{10} = 2^{20}$ file blocks. This more than covers the remaining blocks of our file.
- In the worst case, to access a block of the file we'll have to use the 2-level index. That will involve 3 file accesses in total: one to get to the root, one to the 2nd level, and one to the file itself.

Part E [3 marks in total]

The following program runs without crashing. State the output.

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

int main (void) {

     char * s1 = (char *) malloc(10 * sizeof(char));

     s1[0] = 'A';
     s1[1] = 'B';
     s1[2] = 'C';
     s1[3] = 0;   // end of string indicator.

     char * s2 = (char *) malloc(10 * sizeof(char));
     s2 = s1;
     s2[0] = 'D';
     s2[1] = 'E';

     char * s3;
     s3 = s2;
     s3[0] = 'X';

     // Print the 3 strings.
     cout << s1 << "\n" << s2 << "\n" << s3 << "\n";
}
```

OUTPUT:

```
     XEC
     XEC
     XEC
```

Part F [3 marks in total]

1. Suppose you have a list of UofT student numbers (9 digits long) to write to a file. Which of the following statements is true? Circle one.

   (a) $\boxed{\text{The file will take less disk space if written in binary format.}}$
   (If we assume `sizeof(int) = 4`.)

   (b) The file will take less disk space if written in text format.

   (c) The file will take the same amount of space regardless.

   (d) $\boxed{\text{There is not enough information here to determine which file format will take less space.}}$
   (If we don't assume anything about `sizeof(int)`.)

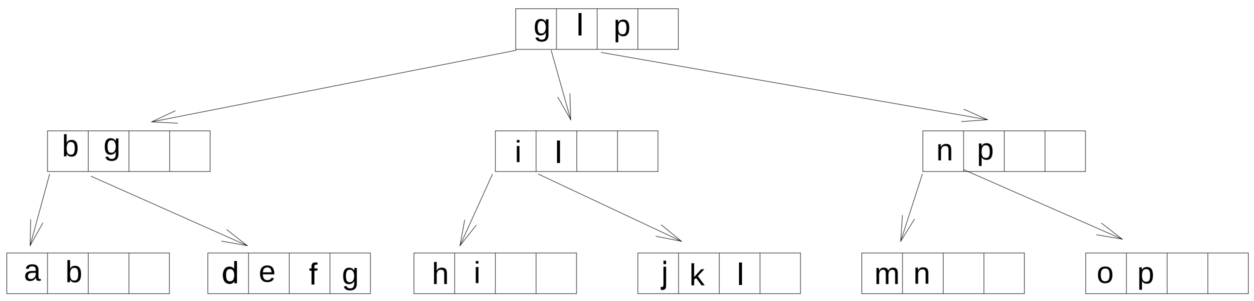
2. Fill in the line of code below so that it writes integer `num` to file stream `fs` in binary format:

```
          fs.write(  &num,   sizeof(int)                          );
```

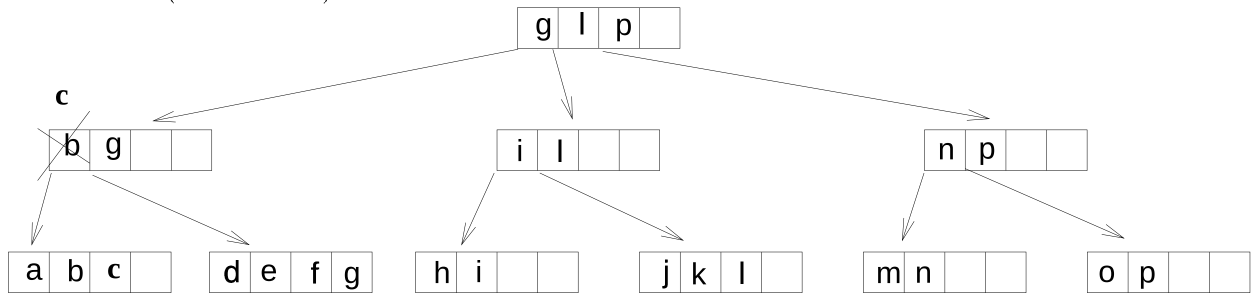
Part G [9 marks in total]

Suppose we have a B-tree with order 4 (*i.e.*, $M = 4$).

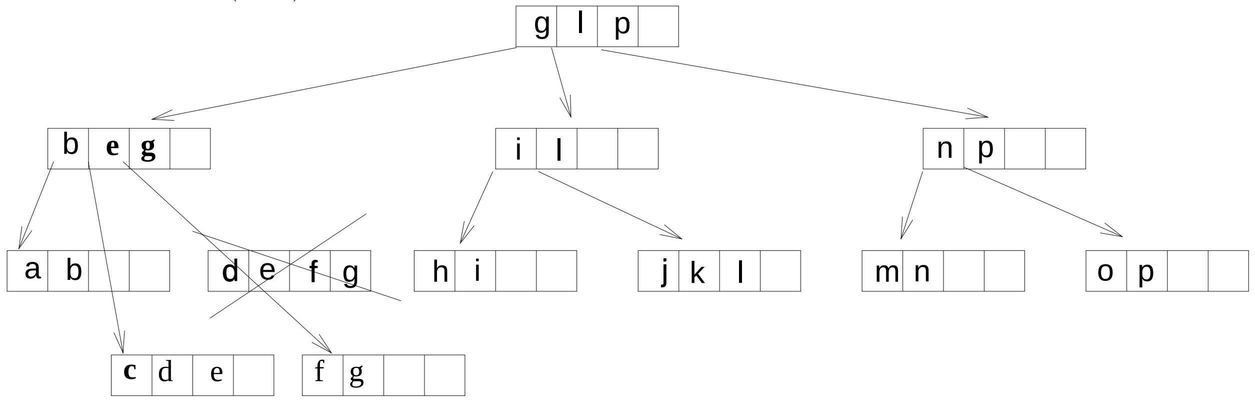
1. Write on the B-tree below to show what would happen if you inserted the key `c`.



One solution (redistribute):

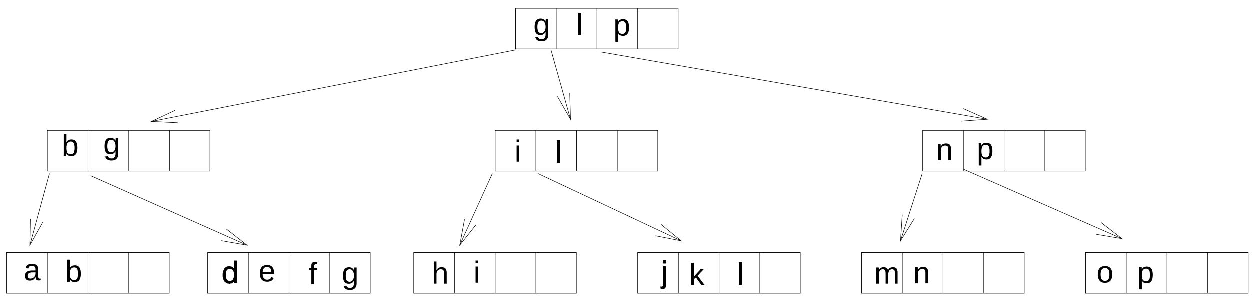


Another solution (split):

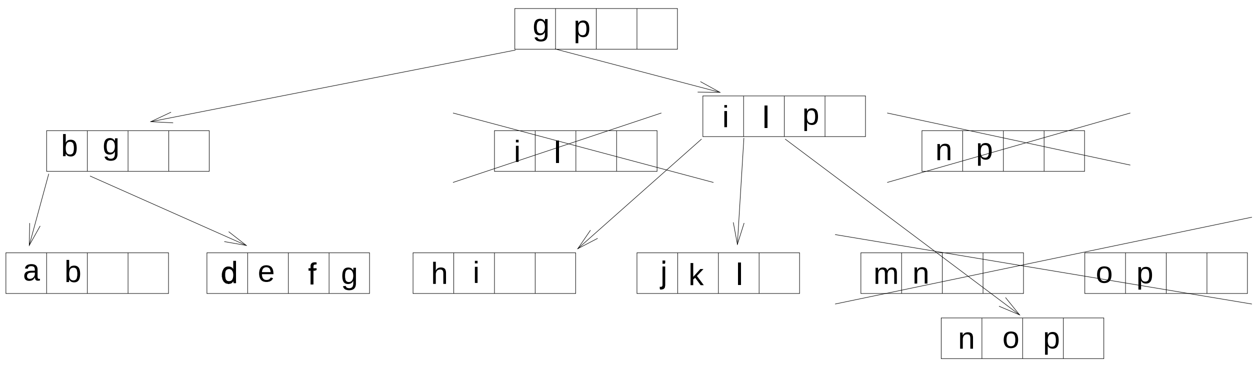


Node splits into two

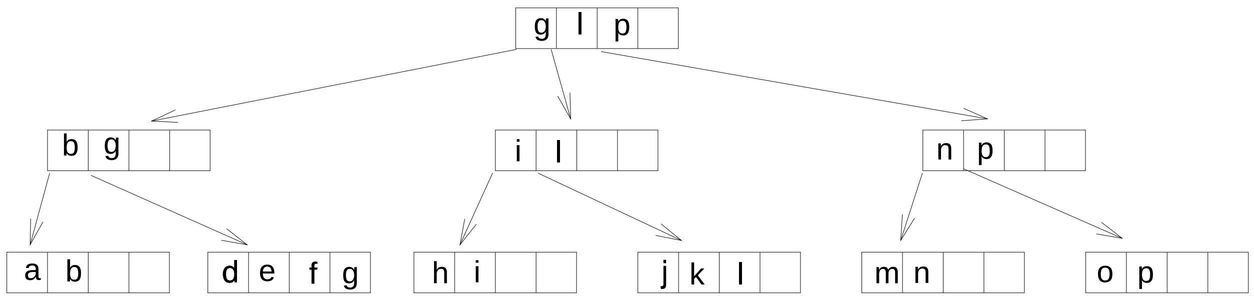
2. On this second copy of the same tree, show what would happen if you deleted m.

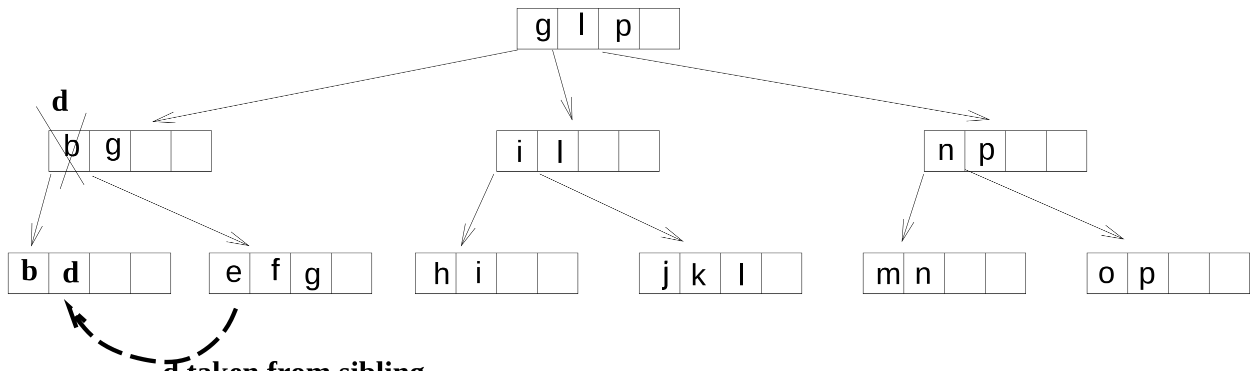


Note: The "l" is gone from the root



3. On this third copy, show what would happen if you deleted a.





d taken from sibling

End of Examination