1. **(5 pts) Problem 4.10.**

**10. Consider the directory tree of Fig. 4-8. If /usr/jim is the working directory, what is the**

**absolute path name for the file whose relative path name is ../ast/x?**

The .. / will move the directory up to the previous one, so the path will be /usr/ast/x.

**(10 pts) Problem 4.13.**

**13. One way to use contiguous allocation of the disk and not suffer from holes is to compact the disk every time a file is removed. Since all files are contiguous, copying a file requires a seek and rotational delay to read the file, followed by the transfer at full speed. Writing the file back requires the same work. Assuming a seek time of 5 msec, a rotational delay of 4 msec, a transfer rate of 80 MB/sec, and an average file size of 8 KB, how long does it take to read a file into main memory and then write it back to the disk at a new location? Using these numbers, how long would it take to compact half of a 16-GB disk?**

Seek = 5 msec

Rotational Delay = 4 msec

Transfer rate = 80 MB/sec

Average size = 8 KB

(5 msec + 4 msec) = 9 msec

16 GB/2 = 8GB

Time to read file:

8 KB file size

8 x 210

23 x 210

23+10 =213 bytes size

80 MB transfer rate

80 x 220

8 x 10 x 220

23 x 220 x 10

23+20 = 223 x 10 bytes/sec

Time to read:

=0.000097656

=0.000097656 x 10 x 102

=0.09776 msec

Total time = 5 + 4+ 0.09776 = 9.0977

Copying a file: 9.0977\*2 = 18.1954msec

8 GB disk store 220 files.

1048.576\*18.1954 = 19079.25 sec = about 5 hours

**(5 pts) Problem 4.14.**

**14. In light of the answer to the previous question, does compacting the disk ever make**

**any sense?**

In the previous case, it does not make sense because compacting the disk after each file removal will take a lot of time. This does not mean that compacting the disk is not useful in general. Periodically compacting the disk can be useful for allowing the user to access files quickly.

**(10 pts) Problem 4.25.**

**25. The beginning of a free-space bitmap looks like this after the disk partition is first formatted:**

**1000 0000 0000 0000 (the first block is used by the root directory). The system**

**always searches for free blocks starting at the lowest-numbered block, so after**

**writing file A, which uses six blocks, the bitmap looks like this: 1111 1110 0000 0000.**

**Show the bitmap after each of the following additional actions:**

1. File B is written, using five blocks.

1111 1111 1111 0000

1. File A is deleted.

1000 0011 1111 0000

1. File C is written, using eight blocks.

1111 1111 1111 1111 1100

1. File B is deleted.

1111 1110 0000 1100

**(10 pts) Problem 5.14.**

**14. In which of the four I/O software layers is each of the following done.**

**(a) Computing the track, sector, and head for a disk read.**

Device driver

**(b) Writing commands to the device registers.**

Device driver

**(c) Checking to see if the user is permitted to use the device.**

Operating system

**(d) Converting binary integers to ASCII for printing**

User-level software

**(10 pts) Problem 5.17.**

**17. How much cylinder skew is needed for a 7200-RPM disk with a track-to-track seek**

**time of 1 msec? The disk has 200 sectors of 512 bytes each on each track.**

7200 RPM/60sec = 120 RPS

1 rotation = 1000/120msec

(1000/120)/200 sectors = 0.04167 or 1/24 msec

Cylinder skew = 24 passed per msec

**(10 pts) Problem 5.19.**

**19. Calculate the maximum data rate in bytes/sec for the disk described in the previous**

**problem.**

120 rotations per second

The disk can read 500x512 bytes in a rotation. 500x512=256,000

In one second the disc can read 256,000x120 = 30,720,000

**(25 pts) Problem 4.43.**

**43. Write a program that reverses the bytes of a file, so that the last byte is now first and**

**the first byte is now last. It must work with an arbitrarily long file, but try to make it**

**reasonably efficient.**

In unzipped folder.

**(25 pts) Problem 4.44.**

**44. Write a program that starts at a given directory and descends the file tree from that**

**point recording the sizes of all the files it finds. When it is all done, it should print a**

**histogram of the file sizes using a bin width specified as a parameter (e.g., with 1024,**

**file sizes of 0 to 1023 go in one bin, 1024 to 2047 go in the next bin, etc.).**

In unzipped folder.