CMR INSTITUTE OF TECHNOLOGY

Answer Scheme and solution

Sub:	File structures					Code:10IS63
Max Marks:	50	Sem:	VI	Branch:	ISE	

		Marks
1 (a)	Differentiate between the physical file and the logical file	[4]

Physical file	Logical file
1. Occupies the portion of memory. It contains the original data.	1. Does not occupy any memory space. Does not contain any data. It loads itself at run time as per the defined access path.
2. A physical file contains one record format	2. A logical file can contain up to 32 record formats.
3.Can exist even without LF	3. Can't exist without PF
4. If there is a logical file for a PF, the PF can't be deleted until and unless we delete the LF.	4. If there is a logical file for a PF, the LF can be deleted without deleting the PF.
5.CRTPF command is used to create such object	CRTLF command is used to create such type object

(b) Summarize the notes about the CD-ROM strengthens and weakness [6]

CD-ROM Strengths & Weaknesses

- Seek Performance: very bad
- Data Transfer Rate: Not Terrible/Not Great
- Storage Capacity: Great
  - Benefit: enables us to build indexes and other support structures that can help overcome some of the limitations associated with CD-ROM's poor performance.

	• Read-Only Access: There can't be any changes ==> File organization can be optimized. • No need for interaction with the user (which requires a quick response)	
2 (a)	Difference between the disk and magnetic tape Magnetic Tapes A magnetic tape is a thin and a long plastic strip coated with a magnetizable material. The recorder orders the magnetizable material on the magnetic tape according to the incoming signal. The reading process is simply done by sending the tape near a coil which produces a current which can be decoded to the original source. Magnetic tapes are also used as computer data storage. These were used before hard disc drives were invented. Magnetic tapes are still used to archive large amounts of data for non-frequent usage. The magnetic tape is a sequential storage device. The data can only be read as a serial input. Magnetic tapes are mostly used in Audio cassettes and video cassettes. Magnetic tapes are used as digital data storage devices as well as analog data storage devices. Magnetic Disks A magnetic disk operates the same way a magnetic tape does, but magnetic disks can usually store a large amount of data than the magnetic tapes. The main advantage of the magnetic disk is that data can be read from anywhere. A magnetic disk is also more portable than the magnetic tape. Computer hard disc drives are the main devices that use magnetic disks. Magnetic disks are not shockproof. A shock can change the current magnetic condition of a material. However, since magnetic tapes are not solid, the chance of a shock is minimal. Magnetic disks are used as digital data storage devices rather than analog data storage devices. A certain area on the disk is known as a block. The net magnetic orientation of a block decides whether it is a digital 0 or a 1.	[5]
b)	Difference between CLV and CAV in detail In optical storage, constant linear velocity (CLV) is a qualifier for the rated speed of an optical disc drive, and may also be applied to the writing speed of recordable discs. CLV implies that the angular velocity (i.e. rpm) varies during an operation, as contrasted with CAV modes. CLV = Constant Linear Velocity CAV = Constant Angular Velocity Pros: CLV: The laser sees the disc moving at the same speed throughout the whole burning session therefore the interection between laser and disk is consistent. CAV: Spindle speed is kept constant while the write speed constantly changes. Cons: CLV: varying RPM will cause the disc to vibrate quite a bit, this can cause write quality errors.	[5]

	Limited to speeds of 16x on drives. CAV: The interection between disk and laser can't be predicted exactly as the velocity of the disk and laser power must be changed continuously.	
3 (a)	It is needed to store a backup of a large file with 1 million 100 bytes of records on a 6250 bpi tape that has an inter block gap of 0.3 inches with a blocking factor of 50, then calculate tape length required Example: – one million 100-byte records – 6,250 BPI tape – 0.3 inches of interblock gap How much tape is needed? – when blocking factor is between 1 and 50 Nominal recording density Effective recording density : – number of byte per block / number of inches for block	[8]
(b)	Define the inter block gap. Why inter block gap has been provided on tape ? *Interblock gap* definition, the area or space separating consecutive blocks of data or consecutive physical records on an external storage medium	[2]
4	Understand the concepts of records creation and write how to Add a structure to the records using relevant example.	[10]
5	Define a file? List the fundamental file operations. Opening Files Opening a file makes it ready for use by the program. Two options for opening a file : • open an existing file • create a new file When we open a file we are positioned at the beginning of the file. In C : . .	[10]

.
FILE * outfile;
outfile = fopen("myfile.txt", "w");
.
.
.
The first argument indicates the physical name of the file. The second one determines the “mode”, i.e. the way, the file is opened.
For example :
“r” = open for reading,
“w” = open for writing (file need not to exist),
“a” = open for appending (file need not to exist),
among other modes (“r+”,“w+”, “a+”).
In C++ :
.
.
.
fstream outfile;
outfile.open("myfile.txt",ios::out);
.
.
.
The second argument is an integer indicating the mode. Its value is set as a “bitwise or” of constants defines in class
Ios
Closing Files
This is like “hanging up” the line connected to a file.
After closing a file, the logical name is free to be associated to another physical file.
Closing a file used for output guarantees everything has been written to the physical file.
We will see later that bytes are not sent directly to the physical file one by one; they are first stored in a buffer to be written later as a block of data.
When the file is closed
the leftover from the buffer is flushed to the file.
Files are usually closed automatically by the operating system at the end of program’s execution.
It’s better to close the file to prevent data loss in case the program does not terminate normally.
In C :
fclose(outfile);
In C++ :
outfile.close();
Reading
Read data from a file and place it in a variable inside the program.
Generic
Read
function (not specific to any programming language)
Read(Source_file, Destination_addr, Size)

Source
file
= logical name of a file which has been opened
Destination
addr
= first address of the memory block were data should
be stored
Size
= number of bytes to be read
In C (or in C++ using C streams) :
char c;
FILE * infile;
.
.
.
infile = fopen("myfile,"r");
fread(&c,1,1,infile);
1st argument:
destination address (address of variable
c
)
2nd argument: element size in bytes (a
char
occupies 1 byte)
3rd argument:
number of elements
4th argument:
logical file name
In C++ :
char c;
fstream infile;
infile.open("myfile.txt",ios::in);
infile >> c;
Note that in the C++ version, the operator
>>
communicates the same info
at a higher level. Since
c
is a char variable, it's implicit that only 1 byte is
to be transferred.
Writing
Write data from a variable inside the program into the file.
Generic
Write
function :
Write (Destination_File, Source_addr, Size)
Destination
file
= logical file name of a file which has been opened

	Source
	addr
	= first address of the memory block where data
	is stored
	Size
	= number of bytes to be written
	In C (or in C++ using C streams) :
	char c;
	FILE * outfile;
	outfile = fopen("mynew.txt","w");
	fwrite(&c,1,1,outfile);
	In C++ :
	char c;
	fstream outfile;
	outfile.open("mynew.txt",ios::out);
	outfile << c;
	Detecting End-of-File
	When we try to read and the file has ended, the read was unsuccessful. We
	can test whether this happened in the following ways :
	In C : Check whether
	fread
	returned value 0
	int i;
	i = fread(&c,1,1,infile);
	if (i==0) // file has ended
	...
	in C++: Check whether
	infile.fail()
	returns
	true
	infile >> c;
	if (infile.fail()) // file has ended
	..	

6	Apply the direct access approach in files to access the records and explain using relevant Example Direct access can also be called *random* access, because it allows equally easy and fast access to any randomly selected destination. Somewhat like traveling by a *Star Trek* transporter instead of driving along the freeway and passing the exits one at a time, which is what you get with sequential access.) In a normal, physical book, the reader is supposed to read pages one by one, in the order in which they are provided by the author. For most books (fiction, at least), it makes little sense for the reader to turn directly page 256 and start reading there. Unless, of course, that is where the reader left off in their last reading session. Getting to page 256 in a 500-pages book poses a bit of a challenge, as we well know it, and each of us have their preferred method of dealing with it (be it	[10]

	a bookmark, a dog ear, or our own memory). Tables of contents try to alleviate a book's sequential-access problem by telling people what content is going to be found in the book and at which page. The user still has the problem of turning to the desired page number, but at least he doesn't need to bother with parsing the content and deciding whether he's found what he is looking for. By definition, however, the web embraces direct access. Thus, it is disappointing to see sequential-access designs becoming increasingly popular nowadays.	

7. Define Buffer Management in files and Explain buffering management Strategies [10]

Buffering means working with large chunks of data in main memory so the number of accesses to secondary storage is reduced.

►

Today, we'll discuss the System I/O buffers. These are beyond the control of application programs and are manipulated by the O.S.

►

Note that the application program may implement its own "buffer" – i.e. a place in memory (variable, object) that accumulates large chunks of data to be later written to disk as a chunk.

Double Buffering:

Two buffers can be used to allow processing and I/O to overlap.

– Suppose that a program is only writing to a disk.

– CPU wants to fill a buffer at the same time that I/O is being performed.

– If two buffers are used and I/O-CPU overlapping is permitted, CPU can be filling one buffer while the other buffer is being transmitted to disk.

– When both tasks are finished, the roles of the buffers can be exchanged.

►

The actual management is done by the O.S.

Mult iple Buffering

: instead of two buffers any number of buffers can be used to allow processing and I/O to overlap.

►

Buffer pooling

:

– There is a pool of buffers.

– When a request for a sector is received, O.S. first looks to see that sector is in some buffer.

– If not there, it brings the sector to some free buffer. If no free buffer exists, it must choose an occupied buffer.
Move mode (using both system buffer & program buffer)
– moving data from one place in RAM to another before they can be accessed
– sometimes, unnecessary data moves
►
Locate mode (using system buffer only or program buffer only)
– perform I/O directly
between secondary storage and
program buffer (program's data area)
– system buffers handle all I/Os, but program uses
locations
through pointer variable