CMR
INSTITUTE OF
TECHNOLOGY

USN | | | | | | | | | | |

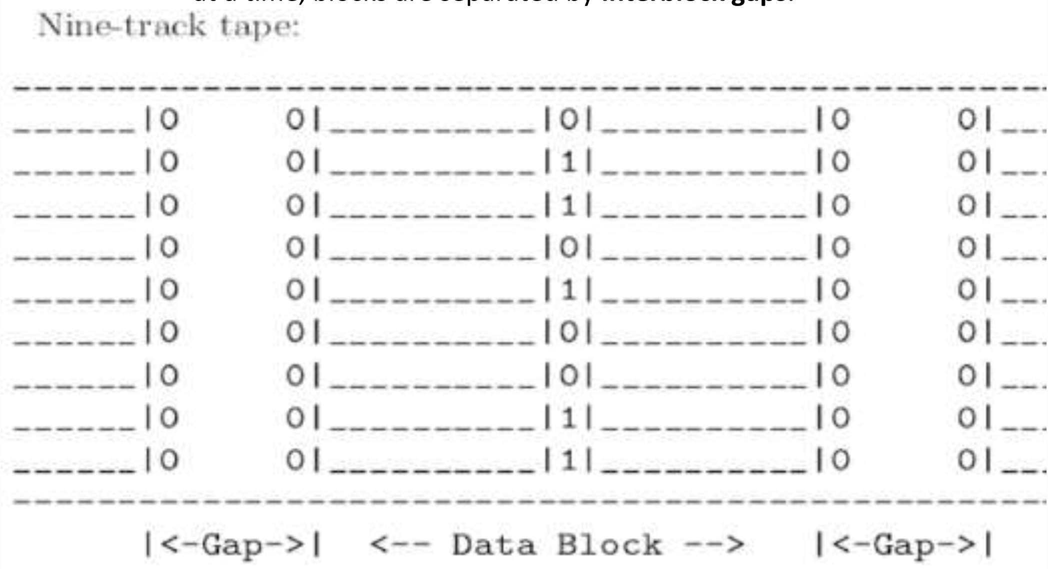
First Internal Test

Sub:	**File Structures**							Code:	15IS62
Date:	05/ 03 / 2019	Duration:	90 mins	Max Marks:	50	Sem:	VI	Branch:	**ISE**
Answer Any **FIVE FULL** Questions									

		Marks	OBE	
			CO	RBT
1 (a)	Explain the organization of data on a nine track tapes with a neat diagram. Suppose it is needed to store a backup of a large file with 1 million records of 100 bytes records on a 7250bpi (bytes per inch) tape that has an internal gap of 0.2 inch and with a blocking factor (records/block) of 60.Hence calculate the length of tape required	[10]	CO1	L3

Ans

- Surface of a typical tape is a set of parallel tracks, each of which is a sequence of bits.
- If there are nine tracks, the nine bits that are at corresponding position in the nine respective tracks which constitutes **a byte of data and 1 parity bit**.
- Such one-bit-wide slice of a tape is known as a **frame**.
- Frames are grouped together into **datablocks**, tapes are read one block at a time, blocks are separated by **interblock gaps**.



Number of blocks = 1000000/60 = 16,667
Each block has 60 x 100 = 6000 bytes
Length of each block = 6000/7250 = 0.8275 inches
Total length per block = 0.8275 + 0.2 = 1.0275”
File size = 16667 x 1.0275 = 17126.78”

2 (a)	Explain the different methods of adding structures to files to maintain the identity of fields and records.	[10]	CO2	L2

Ans

- A record can be defined as a set of fields that belong together when the file is viewed in terms of a higher level of organization.
- Like the notion of a field, a record is another conceptual tool which needs not exist in the file in any physical sense.

- Yet, they are an important logical notion included in the file’s structure.
- Methods for organizing the records of a file include:
    a. Requiring that the records be a predictable number of bytes in length.
    b. fixed-length records
    c. Requiring that the records be a predictable number of fields in length.
    d. Beginning each record with a length indicator consisting of a count of the number of bytes (or number of fields) that the record contains.
    e. Using a second file to keep track of the beginning byte address for each record.

3 (a)	Explain the file operations OPEN, CLOSE, READ, WRITE and SEEK using C++. With suitable example.	[10]	CO2	L2

C++. With suitable example.

OPEN:

A file must be opened before you can read from it or write to it. Either the ofstream or fstream object may be used to open a file for writing and ifstream object is used to open a file for reading purpose only.

Following is the standard syntax for open() function, which is a member of fstream, ifstream, and ofstream objects.

void open(const char *filename, ios::openmode mode);

Here, the first argument specifies the name and location of the file to be opened and the second argument of the open() member function defines the mode in which the file should be opened.

Mode Flag Description

Mode Flag	Description
ios::app	Append mode. All output to that file to be appen end.
ios::ate	Open a file for output and move the read/write c the end of the file.
ios::in	Open a file for reading.
ios::out	Open a file for writing.
ios::trunc	If the file already exists, its contents will be trun before opening the file.

One can combine two or more of these values by ORing them together. For example if you want to open a file in write mode and want to truncate it in case it already exists, following will be the syntax:

```
ofstream outfile;
outfile.open("file.dat", ios::out | ios::trunc );
```

Similar way, you can open a file for reading and writing purpose as follows:

```
fstream  afile;
afile.open("file.dat", ios::out | ios::in );
```

Close:
When a C++ program terminates it automatically closes flushes all the streams, release all the allocated memory and close all the opened files. But it is always a good practice that a programmer should close all the opened files before program termination.

Following is the standard syntax for close() function, which is a member of fstream, ifstream, and ofstream objects.

```
afile.close();
```

Write:
While doing C++ programming, you write information to a file from your program using the stream insertion operator (<<) just as you use that operator to output information to the screen. The only difference is that you use an ofstream or fstream object instead of the cout object.

Read:
You read information from a file into your program using the stream extraction operator (>>) just as you use that operator to input information from the keyboard. The only difference is that you use an ifstream or fstream object instead of the cin object.

4 (a)	Explain the sector based data organization in magnetic disk with a neat diagram. Explain the different costs of disk access and formulate access time for random access and sequential access using the same.	[10]	CO1	L2

Ans The information on disk is stored on the surface of 1 or more platters.

- The information is stored in successive tracks on the surface of the disk.
- Each track is divided into sectors.
- A sector is the smallest addressable portion of a disk.
- Disk drives have a number of platters.
- The tracks directly above one another form a cylinder.
- All information on a single cylinder can be accessed without moving the arm that holds the read/write heads.
- Moving this arm is called seeking.

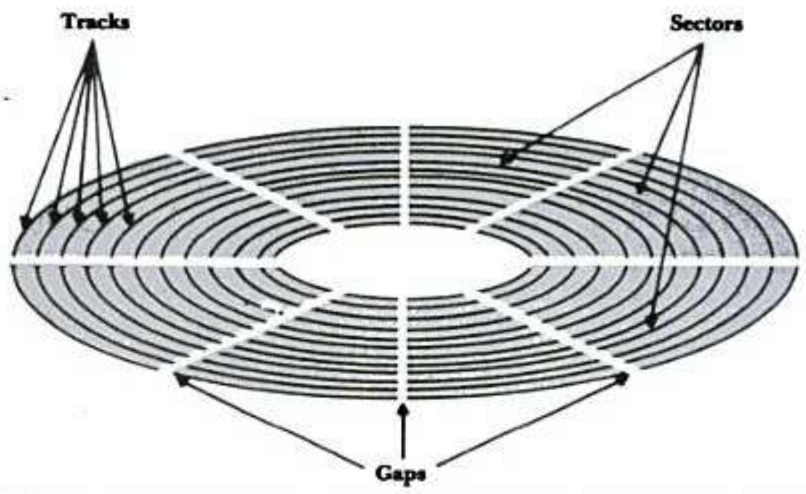


Figure 3.2 Surface of disk showing tracks and sectors.

5 (a)	What is data compression? Explain briefly different types of data compression methods.	[10]	CO1	L2

Ans **Compact Notation**

The replacement of field values with an ordinal number which index an enumeration of possible field values.

∉ Compact notation can be used for fields which have an effectively fixed range of values.

∉ Compact notation can be used for fields which have an effectively fixed range of values. The *State* field of the *Person*record, as used earler, is an example of such a field. There are 676 (26 x 26) possible two letter abbreviations, but there are only 50 states. By assigning an ordinal number to each state, and storing the code as a one byte binary number, the field size is reduced by 50 percent.

∉ No information has been lost in the process. The compression can be completely reversed, replacing the numeric code with the two letter abbreviation when the file is read. Compact notation is an example of redundancy reduction.

On the other hand, programs which access the compressed data will need additional code to compress and expand the data.

Run Length Encoding

An encoding scheme which replaces runs of a single symbol with the symbol and a repetition factor. Run-length encoding is useful only when the text contains long runs of a single value. Run-length encoding is useful for images which contain solid color areas. Run-length encoding may be useful for text which contains strings of blanks.

Example:

uncompressed text (hexadecimal format):
40 40 40 40 40 40 43 43 41 41 41 41 41 42
compressed text (hexadecimal format):
FE 06 40 43 43 FE 05 41 42
where FE is the compression escape code, followed by a length byte, and the byte to be repeated.

variable length encoding

An encoding scheme in which the codes for differenct symbols may be of different length.

huffman code

A variable length code, in which each code is determined by the occurence frequency of the corresponding symbol

6 (a) Write short notes on data organization of CDROM and list the strengths and weakness of CD-ROM. [10] CO1 L1

- CD-ROMs are stamped from a glass master disk which has a coating that is changed by the laser beam. When the coating is developed, the areas hit by the laser beam turn into **pits** along the track followed by the beam. The smooth unchanged areas between the pits are called **lands**.
- Pits scatter light; lands reflect light.
- 1's are represented by the transition from pit to land and back again. 0's are represented by the amount of time between transitions. The longer between transitions, the more 0s we have.
- There must be at least two 0s between any pair of 1s.
- EFM encoding (Eight to Fourteen Modulations) turns the original 8 bits of data into 14 expanded bits that can be represented in the pits and lands on the disk.
- Since 0s are represented by the length of time between transition, the disk must be rotated at a precise and constant speed. This affects the CD-ROM drive's ability to seek quickly.
- Different from the "regular" disk method.
- Each second of playing time on a CD is divided into 75 sectors. Each sector holds 2 Kilobytes of data. Each CD-ROM contains at least one hour of playing time.

The disc is capable of holding at least 60 min * 60 sec/min * 75 sector/sec * 2 Kilobytes/sector = 540, 000 KBytes

- Often, it is actually possible to store over 600, 000 KBytes.
- Sectors are addressed by min:sec:sector e.g., 16:22:34

7 (a) Explain briefly how to manipulate buffer using classes with any one of the class declaration. [10] CO2 L2

Ans **Goal:** Encapsulate the pack, unpack, read, write operations of buffers

Usage

Output: start with an empty buffer object, pack field values into the object, then write buffer to output stream.

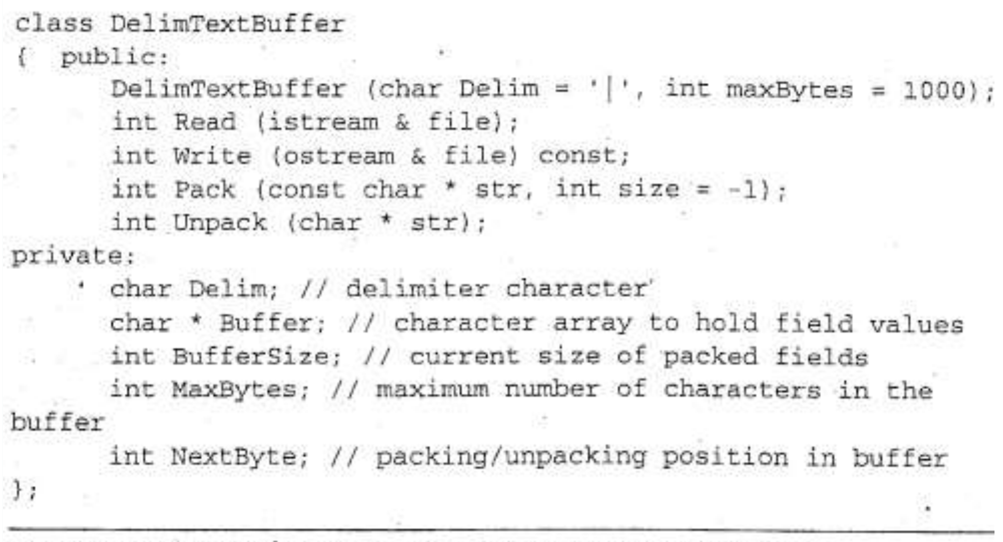
Input: initialize a buffer object by reading a record from input stream, then unpack field values one by one.

Constraints

No updates on packed data

No mixing of pack and unpack operations

Class DelimTextBuffer: For Variable length records with delimited fields.



```
class DelimTextBuffer
{  public:
      DelimTextBuffer (char Delim = '|', int maxBytes = 1000);
      int Read (istream & file);
      int Write (ostream & file) const;
      int Pack (const char * str, int size = -1);
      int Unpack (char * str);
private:
      char Delim; // delimiter character
      char * Buffer; // character array to hold field values
      int BufferSize; // current size of packed fields
      int MaxBytes; // maximum number of characters in the
buffer
      int NextByte; // packing/unpacking position in buffer
};
```

Figure 4.11 Main methods and members of class DelimTextBuffer.

8 (a)	If a file with 50000 fixed length data records has to be stored on a small computer disk with the following characteristics: Number of bytes per sector = 58 Number of sectors per track = 512 Number of tracks per cylinder = 14 Number of cylinders = 3852 How many cylinders does the file require if each data record requires 128 bytes?	[10]	CO1	L3
Ans	Sector capacity=58 Track capacity=58*512 cylinder capacity=14*58*512 Total file size=50000*128 No of cylinders= (50000*128)/(14*58*512)=15.39=16 cylinders			