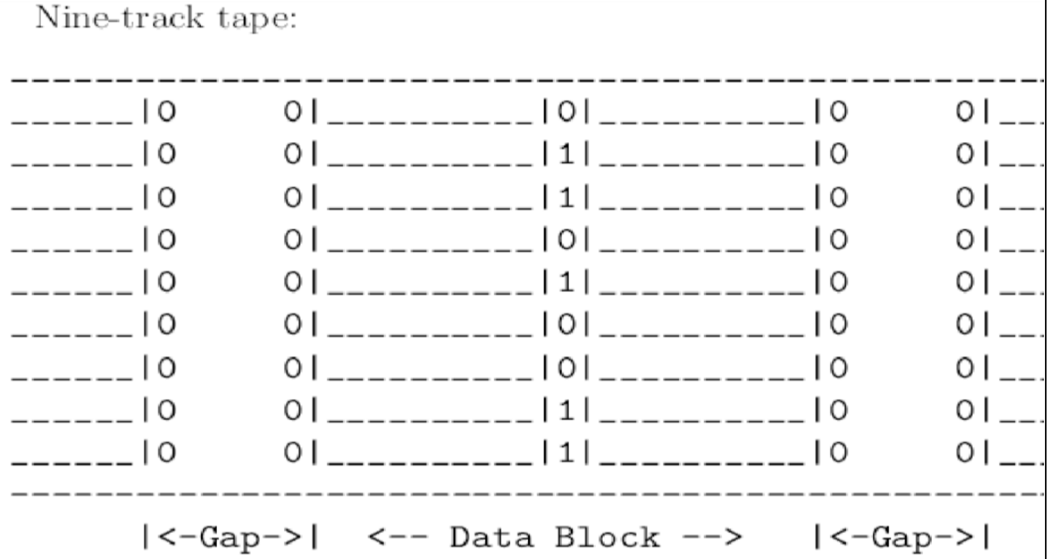
CMR INSTITUTE OF TECHNOLOGY	USN								
First Internal Test									
Sub:	File Structures							Code:	15IS62
Date:	12/ 03 / 2018	Duration:	90 mins	Max Marks:	50	Sem:	VI	Branch:	ISE
Answer Any **FIVE FULL** Questions									

		Marks	OBE	
			CO	RBT
1 (a)	Explain the organization of data on a nine track tapes with a neat diagram.	[5]	CO1	L2

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**.



```
Nine-track tape:

-------------------------------------------------------
______|0      0|___________|0|___________|0      0|__
______|0      0|___________|1|___________|0      0|__
______|0      0|___________|1|___________|0      0|__
______|0      0|___________|0|___________|0      0|__
______|0      0|___________|1|___________|0      0|__
______|0      0|___________|0|___________|0      0|__
______|0      0|___________|0|___________|0      0|__
______|0      0|___________|1|___________|0      0|__
______|0      0|___________|1|___________|0      0|__
-------------------------------------------------------

      |<-Gap->|  <-- Data Block -->   |<-Gap->|
```

		Marks	CO	RBT
(b)	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" and with a blocking factor (records/block) of 60.Hence calculate the length of tape required.	[5]	CO1	L3

Ans

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"

		Marks	CO	RBT
2 (a)	Explain the different methods of adding structures to files to maintain the identity of 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.

Example for each

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

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);

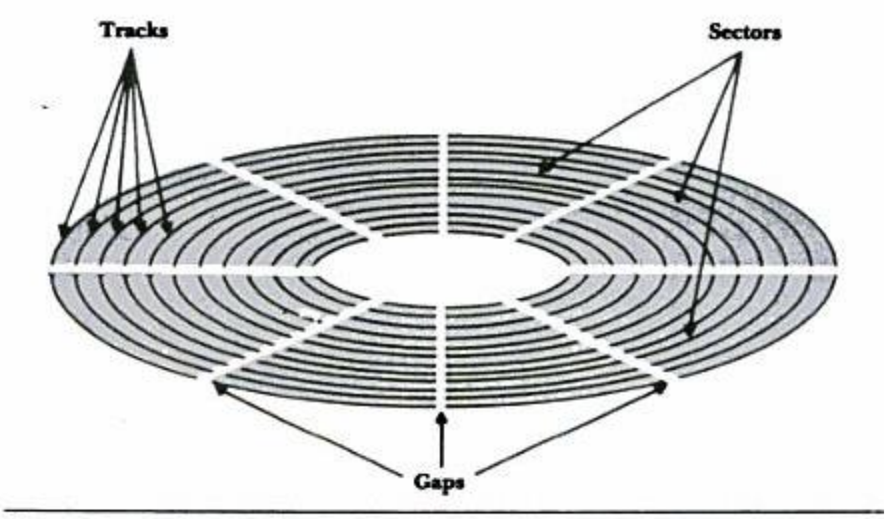
Here, the first argument specifies the name and location of the file to be opened and the second argument of the open() member function defines the mode in which the file should be opened.

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
	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]	CO5	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.

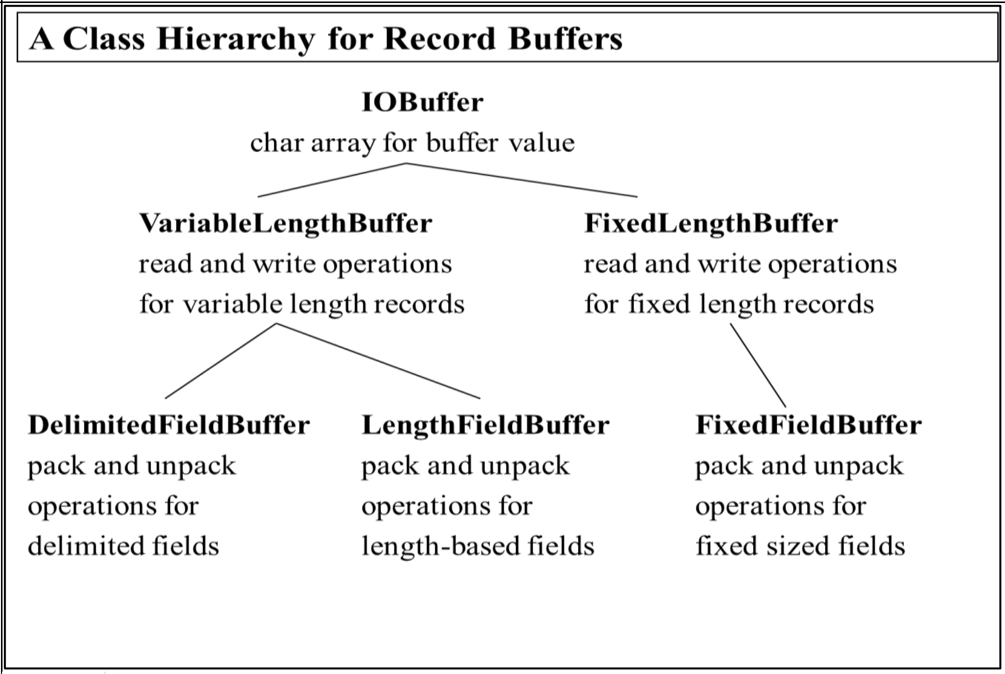
variable length encoding

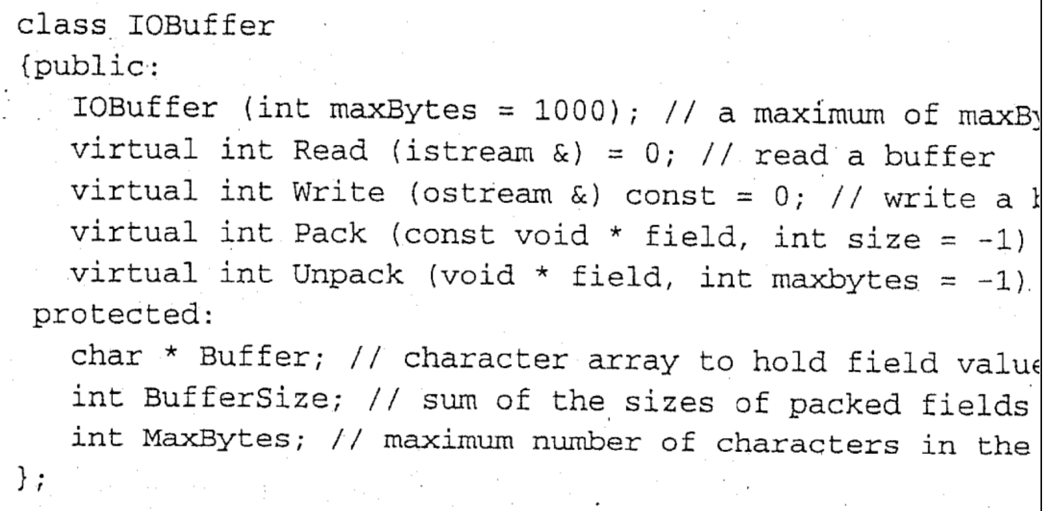
An encoding scheme in which the codes for differenct symbols may be of different length.

huffman code

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

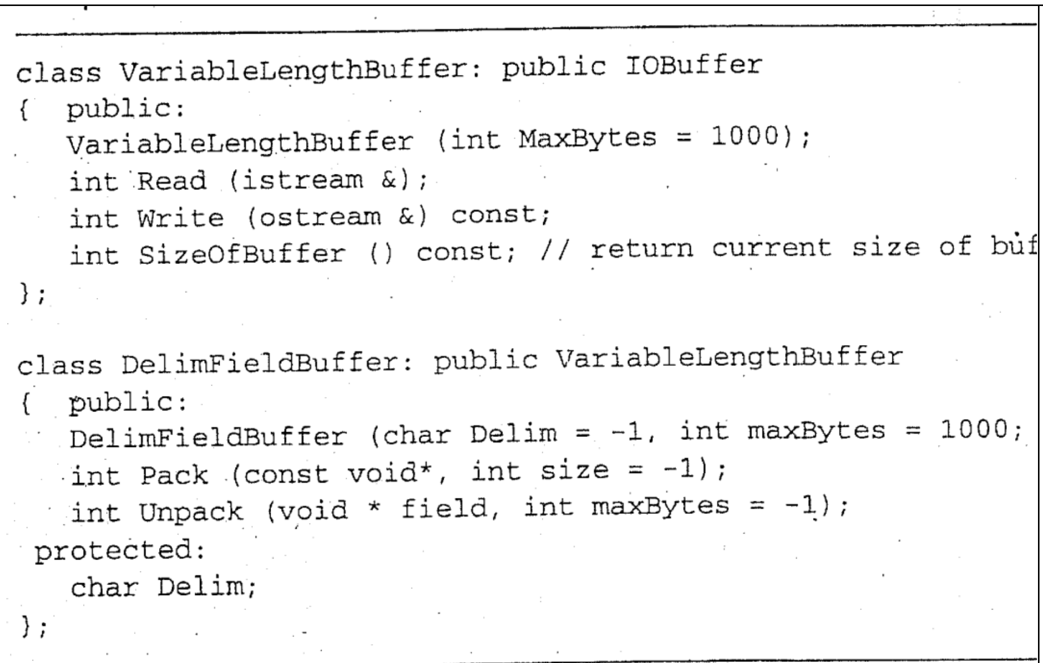
6 (a)	Explain the concept of inheritance using IoBuffer class hierarchy.	[10]	CO2	L2





```
class IOBuffer
{public:
   IOBuffer (int maxBytes = 1000); // a maximum of maxBy
   virtual int Read (istream &) = 0; // read a buffer
   virtual int Write (ostream &) const = 0; // write a b
   virtual int Pack (const void * field, int size = -1)
   virtual int Unpack (void * field, int maxbytes = -1)
 protected:
   char * Buffer; // character array to hold field value
   int BufferSize; // sum of the sizes of packed fields
   int MaxBytes; // maximum number of characters in the
};
```

Figure 4.15 Main members and methods of class IOBuffer.



```
class VariableLengthBuffer: public IOBuffer
{  public:
   VariableLengthBuffer (int MaxBytes = 1000);
   int Read (istream &);
   int Write (ostream &) const;
   int SizeOfBuffer () const; // return current size of bůf
};

class DelimFieldBuffer: public VariableLengthBuffer
{  public:
   DelimFieldBuffer (char Delim = -1, int maxBytes = 1000;
   int Pack (const void*, int size = -1);
   int Unpack (void * field, int maxBytes = -1);
  protected:
   char Delim;
};
```

Figure 4.16 Classes VariableLengthBuffer and DelimFieldBuffer.

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.

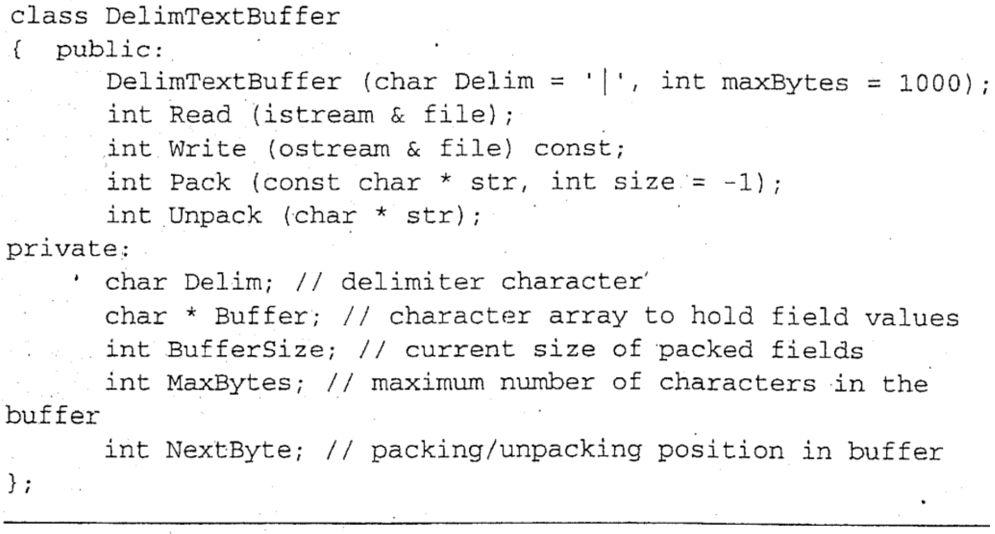
Input: initialize a buffer object by reading a record from input stream, then unpack field values one by one.

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	[10]	CO1	L4

	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?			
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			