I IA

FS

Q) Strength and weakness of CD-ROM.

The strength and weakness of CD-ROM are :-

1) **Seek performance** :-
- The chief weakness of CD-ROM is random access performance.
- Current magnetic disk has an average random access time of 30msec.
On CD-ROM it takes 500 msec.

2) **Data Transfer Rate** :-
- A CD-ROM drive reads seventy sectors or 1540 KB of Data per second.
- 5 times faster than the transfer rate of floppy disk.
- The transfer rate is faster when compared to CD-ROM's seek performance.

3) **Storage capacity** :-
- CD ROM holds more than 600 MB of data.
- With such a large capacity it enables us to build indexes and other structures that can help us to overcome the disadvantages of seek performance of CD-ROM.

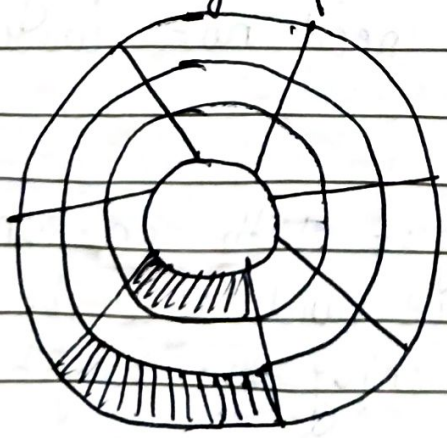
4) **Read only access** :- CD-ROM is a storage device that cannot be changed after developing it.
The advantage is that the user need not worry about updating.

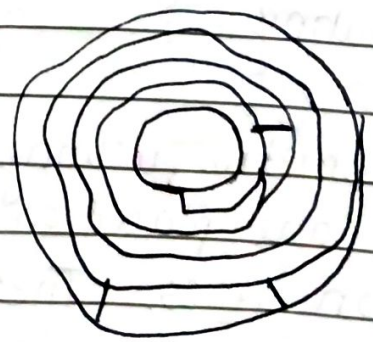
5) **Assymetric writing and reading** :- With CD-ROM we create files are placed on the disk once and access the files 1000 or million times. The user can enjoy the benefit of this investment again and again.

Q) Compare :-

(i) ~~CIV~~ CLV & CAV.

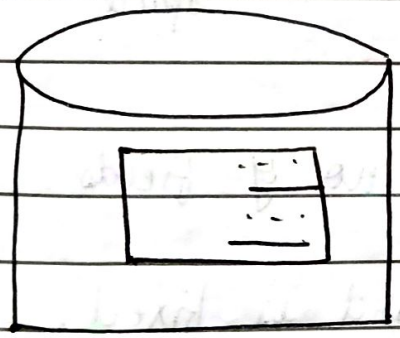
CAV	CLV
• CAV is content angular velocity	• CLV is constant linear velocity.
• Disk is divided into pie shaped sector	• Disk is divided into sectors of equal size.
• Data is densely packed at the centre & loosely packed at the ends.	• Data is densely packed all over the disk.
• Wastage of disk space at outer edge.	• No wastage of space.
• Disk spins at a constant speed	• Disk spins at slower speed when reading at outer edge.
• Lesser storage space	• Much better storage space.



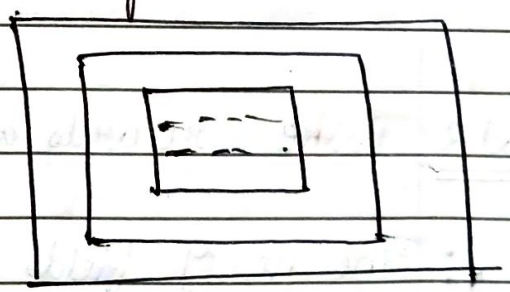


(ii) Physical file and logical file.

Physical file	Logical file.
• Resides in secondary memory	• Resides in primary memory.
• changes are not done	• changes are done directly to logical file by applicat.
• changes are done when file is saved.	• changes are updated when application changes the values.
• can be accessed by any no. of applicants.	• can be accessed by only application which handle it.
• Secondary memo	• Primary memory.



Program data area.



10 marks

Q) Develop a diff record structure used in organization of file.

→ Record structure used in organization of file :-

Method 1: Make records a predictable no of bytes.

- A fixed length record is one in which each record contains the same no. of bytes.

- The record size can be determined by adding the maximum space occupied by each field. This method is also called as counting by tes structure.

Fixed length records with variable length fields :-

AMAR|ID12015061|17|54|13 ← unused space →
MARY|ID12015063|15|14|27 ← unused space →.

Method 2 Make records a predictable no of fields.

- The no of fields in each record is fixed.

- This method is called as fixed field count structure.

Assuming that every record has 5 fields, then each record can be displayed by counting the fields modulo five.

AMAR|ID12015061|17|14|54|MARY|ID12015063|17|13|12.

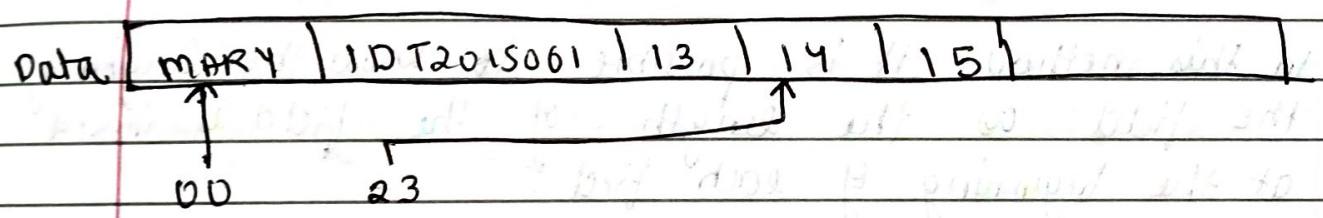
Method 3: Begin each record with a length indicator.

In this method, every record would begin with an integer, that indicates how many bytes are there in the rest of the record.

23 MARY | IDT2015063 | 17 | 13 | 15.

Method 4: Use an index to keep track of addresses.

- An index is used to keep a byte offset for each record.
- The byte offset allows us to find the beginning of each successive record and compute the length of each record.



Method 5: Place a delimiter at the end of each record.

- A special character other than the field delimiter is placed at end of each record, which when encountered indicated the end of the record.

AMAR | IDT2015061 | 14 | 15 | 54 | #.

10marks.

Q) Different ways of adding structures to files.

→ **Method 1:** Fix the length of the fields.

Each variable is stored as an array of characters of fixed length.

eg:- If the name is "Chaya", this occupies only 5 spaces, the rest 5 space is occupied by padding with symbol like #.

eg:-

c	h	a	y	a	\0	#	#	#	#	→
0	1	2	3	4	5	6	7	8	9	next field.

Method 2: Begin each field with a length indicator.

In this method, it is possible to know the length of the field as the length of the field is stored at the beginning of each field.

This type of fields are called length-based field.

eg:-

04 MARY | IDT2015062 | 102 | 202 | 3.

06 AMAR08 | IDT2015061 | 50 | 21 | 70 | 25.

Method 3: Seperate the fields with delimiter.

- Fields are seperated using special characters or sequence of characters.
- A delimiter can be '|', '#', space etc.

eg:- AMAR | 1DT2015063 | 53 | 12 | 21.

<u>Method 4:</u> use a "keyword = value" expression to identify fields.

- In this method, the keyword and its values are stored for each ~~field~~ record.
- The first structure provides info about istself in a field.
- It is easy to identify missing fields.

eg:- Name = AMAR | USN = 1DT2015063 | m1 = 32 | m2 = 6 | m3 = 10

[module 2].

Q) Explain operations required to maintain index files. Define index.

⟹ An index is a table consisting of a list of keys and corresponding reference fields.

OR

An index is a tool for finding records in a file. It consists of :-

> Key field on which the index is searched
> Reference field that tells particular address of the key.

★ operations required to maintain index files :-

> Creating the files :-

Two files must be created :

(i) data field - to hold the data objects.
(ii) index file - to hold the primary key index.

> Loading the index into memory :-

- Only the index file is loaded into the memory.
- Loading of memory is done by sequentially reading a large no. of index records.

> Rewriting the index file from memory :-

- The index file that was loaded on to memory needs to be rewritten back to the index file.
- The changed index file has to be rewritten back to the memory.

> **Record addition :-**

- Adding a new record ~~to~~ data ~~field~~ file ~~that~~ requires that we also add an entry to the index.
- The byte offset of new record should be saved.
- Rearrangement of the index ~~are~~ needs to be done. we have to shift all the records below by one to open up space for inserting the new record.

> **Record ~~deletion~~ deletion :-**

- To delete a record, we just remove the entry from index.
- The space created can be filled up by shifting the below entries to close the space.

> **Record updating :-**

Record updating has two categories :-

- The update changes the value of the key field.
- The update does not affect the key field.

DOMS Page No.
Date / /

[module 2]

Q) Explain index for ~~empty~~ entry seq. file.

[same answer]

[refer previous question].

Q) **Algorithm for key sorting**

1. Read record into buffer
2. Store only the keys in memory.
3. Sort the keys.
4. Create a new file to store the records.
5. Read a record in sorted order of keys.
6. Copy the record to the new file.
7. Continue the step 5 & 6 till the end of sorted key

eg:-

```
int key sort ( fixed record file & infile, char
                    * out file)
  {
    for (int i=0; i<infile : numrecs (); i++)
      {
        infile .readby RRN (obj, i);                    //1
        keynodes [i]= key RRN ( obj, key (), i);        //2
      }
  sort (KEYNODES, infile . numrecs ());                 //3
  Outfile. create (out filename);                       //4
  for (int j=0; j< infile. numrecs ();  j++)
    {
      infile. readby RRN (obj, KEYNODES [j], RRN);      //5
      outfile. APPEND (obj);                            //6
    }
    return -1;
  }.
```

Q) What is file structure?

→ A file structure is the organization of data in files and files in secondary storage and operations for accesing the data.

A file structure allows the applicants to read, write and modify data.

Q) Evolution of file structure

- Earlier files were stored on tapes.
- Access to these tapes was sequential.
- The cost to access the tape increased when the size of file increased.
- As files grew and storage devices were available, ~~indexed~~ indexes were added.
- Index consists of a list of keys and pointers.
- As index file grew, managing became difficult.
- In 1960's, idea of applying ~~tree~~ ~~emerged~~ tree struct emerged.
- But this also resulted in long searches due to uneven growth of trees.
- In 1963, researchers developed an elegant self adjusting binary tree called AVL for storing data.
- The problem with this was that dozen of access was required to find a record.
- The solution emerged in the form of B-tree
- But all the above methods needed more than one disk.
- Hashing approach promised to give required data in

one disk access, but performed badly for large amount of data.

- Extendible Dynamic hashing approach is proposed to improve the performance of hashing algorithm.

Q) Fundamental file processing operations :-

- Opening files
- Closing files
- Reading
- Writing
- Seeking.

[explain].

(20) — I can solve

DOMS Page No.
Date / /

// program in c

```
# include < stdio.h>
main()
{
 char ch;
 FILE * file;
 char filename [20];
 printf (" Enter the name of the file : ");
  gets (filename);
 file = fopen (filename, "r");
while (fread (& ch, 1, 1, file) != 0)
        fwrite (&ch, 1, 1, stdout);
  fclose (file);
}
```

// program in c++

```
  # include <fstream.h>
main () {
 char ch;
 fstream file,
 char filename [20];
cout << " Enter the name of file : "
        << flush;
 cin >> filename;
file.open ( filename ios :: in);
file.unset f ( ios :: skipos);
while (1)
 {
 file >> ch;
 if ( file.fact (1) breaks;
```