Unit - VI

6 Hashing and File Organization

6.1 : Hash Tables and Scattered Tables

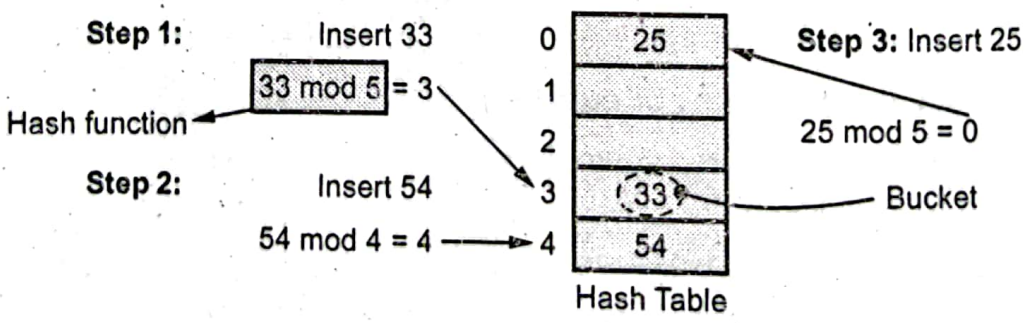
Q.1 Define the terms - 1) Hash Table 2) Hash Function 3) Collision 4) Probe 5) Synonym 6) Overflow 7) Perfect Hash Function

Ans. : 1) Hash table : Hash table is a data structure used for storing and retrieving data quickly. Every entry in the hash table is made using Hash function.

2) Hash function :

- Hash function is a function used to place data in hash table.
- Similarly hash function is used to retrieve data from hash table.
- Thus the use of hash function is to implement hash table.

For example : Consider hash function as **key mod 5**. The hash table of size 5.



3) Collision : Collision is situation in which hash function returns the same address for more than one record.

For example

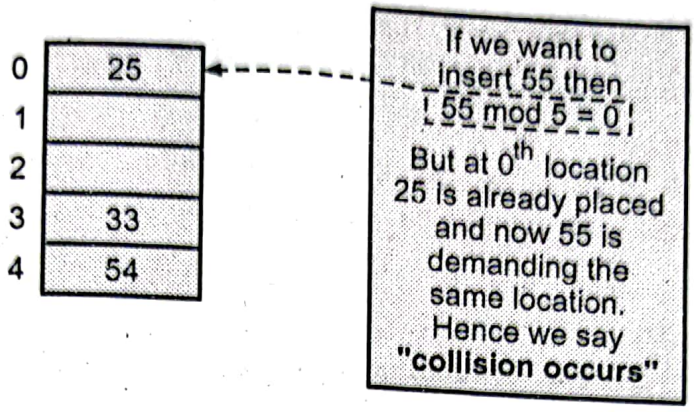


Fig. Q.1.1

4) **Probe :** Each calculation of an address and test for success is known as a probe.

5) **Synonym :** The set of keys that has to the same location are called synonyms. For example - In above given hash table computation 25 and 55 are synonyms.

6) **Overflow :** When hash table becomes full and new record needs to be inserted then it is called overflow.

For example -

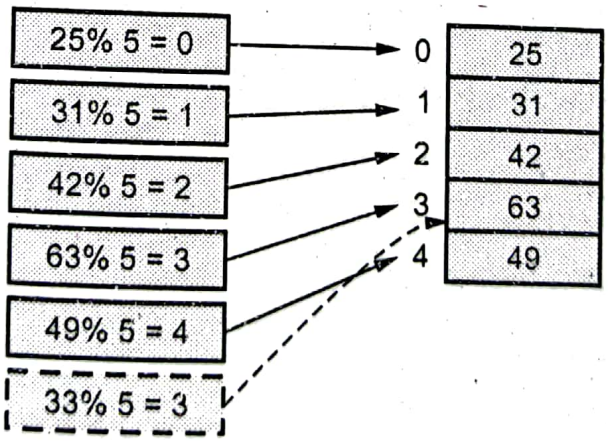


Fig. Q.1.2 Overflow situation

7) **Perfect hash function :** The perfect hash function is a function that maps distinct key elements into the hash table with no collisions.

Q.2 What do you understand by Hash Table ? Why they came into existence ? Discuss different hash functions.

☞ **[SPPU : May-13, Marks 8]**

Ans. : Hash Table - Hash table is a data structure used for storing and retrieving data quickly. Every entry in hash table is made using hash function.

- The hash table is used to get the direct address of the location at which the record can be placed. It is basically used in direct access files.

Hash function - Refer Q.3.

6.2 : Hash Function

Q.3 Define Hash function. What are different Hash functions ?

☞ **[SPPU : June-22, Marks 6]**

Ans. : Hash function : Refer Q.1.

i) Mid square :

- This method works in following steps

1) Square the key

2) Extract middle part of the result. This will indicate the location of the key element in the hash table.

- Note that if the key element is a string then it has to be preprocessed to produce a number.

Let **key = 3111**

$\therefore \quad (3111)^2$

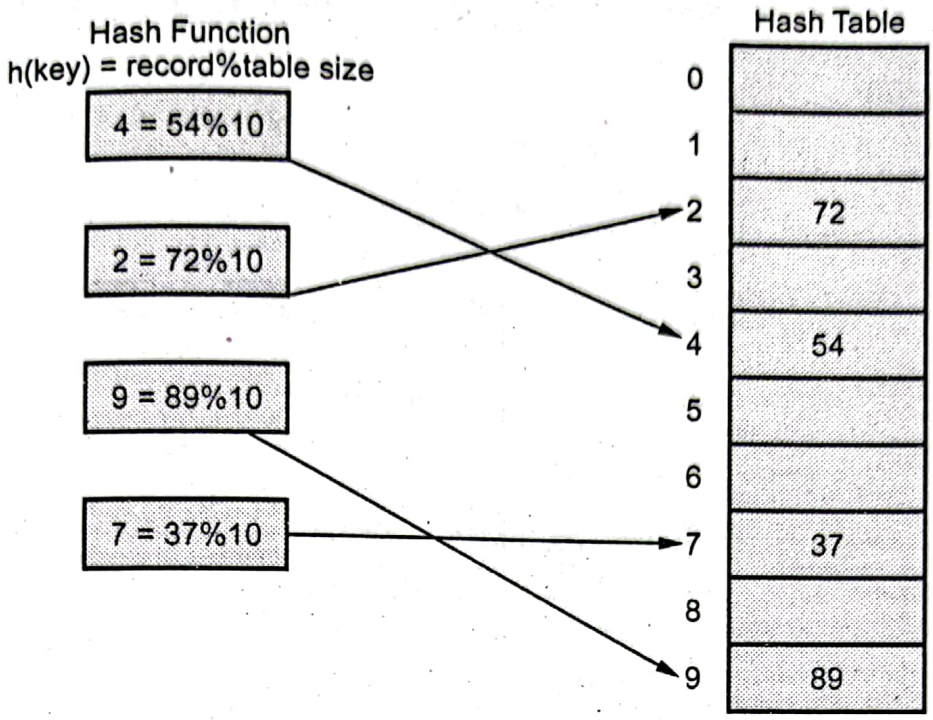
9 6 7 8 3 2 1

- For the hash table of size of 1000

  H(3111) = 783

ii) Modulo division :

- The hash function depends upon the remainder of division.



- Typically the divisor is table length. For example :-
- 'If the record 54, 72, 89, 37 is to be placed in the hash table and if the table size is 10 then

iii) Folding method :

- There are two folding techniques

i) Fold shift ii) Fold boundary

i) **Fold shift :** In this method the key is divided into separate parts whose size matches with the size of required address. Then left and right parts are shifted and added with the middle part.

ii) **Fold boundary :** In this method the key is divided into separate parts. The leftmost and rightmost parts are folded on fixed boundary and added with the middle part.

For example

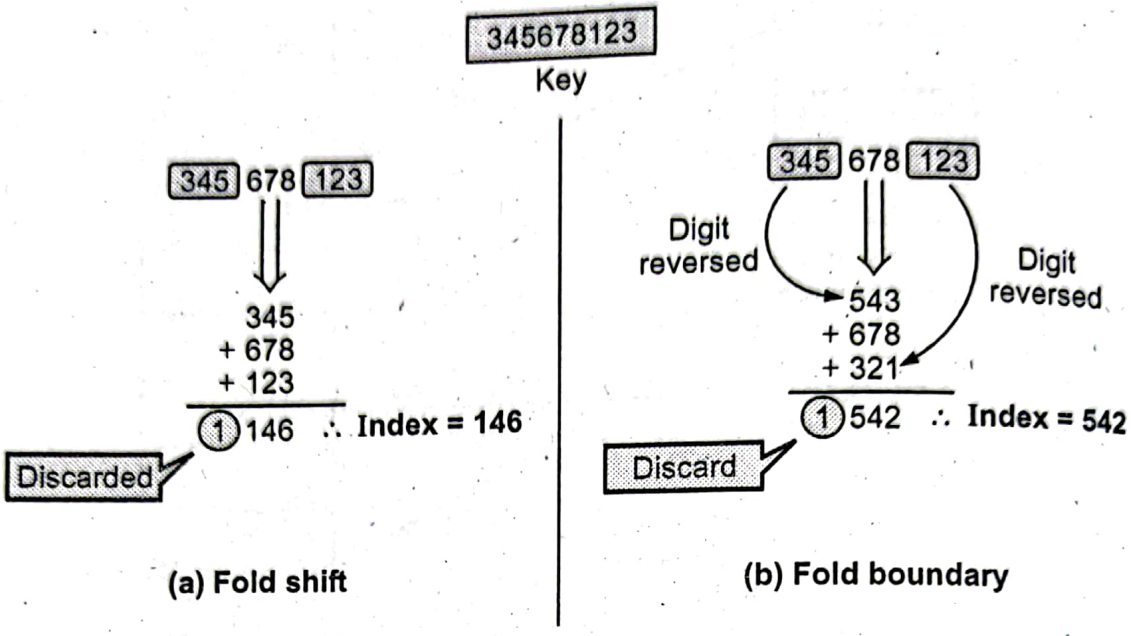
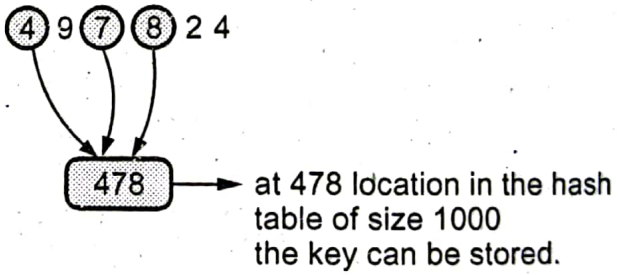


Fig. Q.3.1 Folding techniques

iv) Digit analysis :

- In this method some digits are extracted from the key to form the address location in hash table.



For example : Suppose first, third and fourth digit from left is selected for hash key.

6.3 : Characteristics of Good Hash Function

Q.4 What are the characteristics of good hash function ?

☞[SPPU : June-22, Marks 4]

Ans.: 1. The hash function should be simple to compute.

2. Number of collisions should be less while placing the record in the hash table. Ideally no collision should occur. Such a function is called perfect hash function.

3. Hash function should produce such keys which will get distributed uniformly over an array.

4. The hash function should depend on every bit of the key. Thus the hash function that simply extracts the portion of a key is not suitable.

6.4 : Synonym or Collision

Q.5 Define the term collision. Explain with suitable example.

Ans.:

Definition : The situation in which the hash function returns the same hash key(home bucket) for more than one record is called collision and two same hash keys returned for different records is called synonym.

- Similarly when there is no room for a new pair in the hash table then such a situation is called **overflow**. Sometimes when we handle collision it may lead to overflow conditions. Collision and overflow show the poor hash functions.

For example :

- Consider a hash function.
- H(key) = recordkey%10 having the hash table of size 10.
- The recordkeys to be placed are

131, 44, 43, 78, 19, 36, 57 and 77

Now if we try to place 77 in the hash table then we get the hash key to be 7 and at index 7 already the recordkey 57 is place. This situation is called collision.

0	
1	131
2	
3	43
4	44
5	
6	36
7	57
8	78
9	19

6.5 : Collision Resolution Techniques

Q.6 Enlist various collision resolution techniques.

Ans.: If collision occurs then, colliding records need to be placed at some proper location. This process is called **collision resolution technique.**

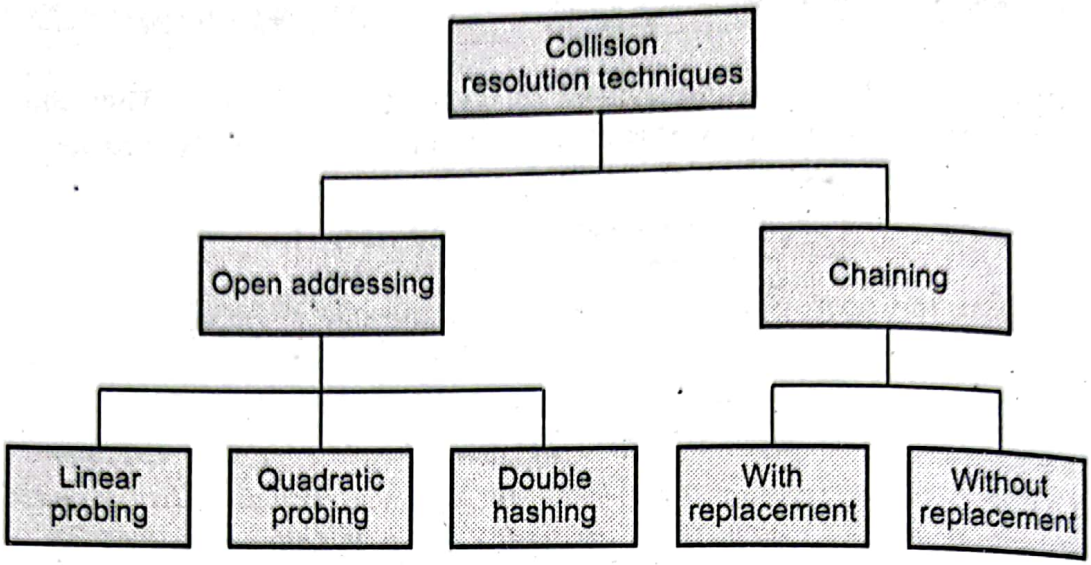


Fig. Q.6.1 Collision resolution technique

Q.7 Explain chaining without replacement and with replacement.

Ans. : Chaining without replacement : A separate chain table is maintained in order to keep track of colliding elements. When collision occurs, then we store the colliding element at next empty location by linearly searching the table. This location normally does not represent the hash key of the colliding element.

- **For example** consider elements,

131, 3, 4, 21, 61, 6, 71, 8, 9

- From the example, From Fig. Q.7.1 that the chain is maintained the number who demands for location 1. First number 131 comes we will place at index 1. Next comes 21 but collision occurs so by linear probing we will place 21 at index 2, and chain is maintained by writing 2 in chain table at index 1 similarly next comes 61 by linear probing we can place 61 at index 5 and chain will be maintained at index 2. Thus any element which gives hash key as 1 will be stored by linear probing at empty location but a chain is maintained so that traversing the hash table will be efficient.

Index	Data	Chain
0	–1	–1
1	131	2
2	21	5
3	3	–1
4	4	–1
5	61	7
6	6	–1
7	71	–1
8	8	–1

Fig. Q.7.1 Chaining without replacement

Chaining with replacement : The colliding element is always probed at the location indicated by the hash key. If that slot is occupied by some another element then that element is always replaced by the colliding element.

For example - Refer Q.13.

Q.8 What is the probing in hash table ? What is probing ? How does it differs from quadratic probing ? Explain with suitable example.

Ans. : Probing is a technique of placing the colliding element in the hash table at the proper empty location.

i) Linear probing : When collision occurs i.e. when two records demand for the same location in the hash table, then the collision can be solved by placing second record linearly down wherever the empty location is found.

For example

Index	data
0	
1	131
2	21
3	31
4	4
5	5
6	61
7	7
8	8
9	

Fig. Q.8.1 Linear probing

In the hash table given in Fig. Q.8.1 the hash function used is number % 10. If the first number which is to be placed is 131 then 131 % 10 = 1 i.e. remainder is 1 so hash key = 1. That means we are supposed to place the record at index 1. Next number is 21 which gives hash key = 1 as 21 % 10 = 1. But already 131 is placed at index 1. That means collision is occurred. We will now apply linear probing. In this method, we will search the place for number 21 from location of 131. In this case we can place 21 at index 2.

2) Quadratic probing : Quadratic probing operates by taking the original hash value and adding successive values of an arbitrary quadratic polynomial to the starting value. This method uses following formula -

$$H_i(key) = (Hash(key)+i^2)\%m$$

- where m can be a table size or any prime number.

For example : If we have to insert following elements in the hash table with table size 10 :

37, 90, 55, 22, 11, 17, 49.

We will fill the hash table step by step

37%10 = 7
90%10 = 0
55%10 = 5
22%10 = 2
11%10 = 1

0	90
1	11
2	22
3	
4	
5	55
6	
7	37
8	
9	

- Now if we want to place 17 a collision will occur as 17%10=7 and bucket 7 has already an element 37. Hence we will apply quadratic probing to insert this record in the hash table.

$$H_i(key) = (Hash(key)+i^2)\%m$$

we will choose value i = 0, 1, 2, ..., whichever is applicable.

Consider i = 0 then

$(17+0^2)\ \%10 = 7$

$(17+1^2)\ \%10 = 8$, when i = 1

- The bucket 8 is empty hence we will place the element at index 8.

0	90
1	11
2	22
3	
4	
5	55
6	
7	37
8	17
9	49

Fig. Q.8.2

- Then comes 49 which will be placed at index 9.

49%10 = 9

Q.9 Explain the need of rehashing with example.

☞[SPPU : Dec.-17, Marks 4]

Ans. : Rehashing is a technique in which the table is resized, i.e., the size of table is doubled by creating a new table. It is preferable if the total size of table is a prime number. There are situations in which the rehashing is required-

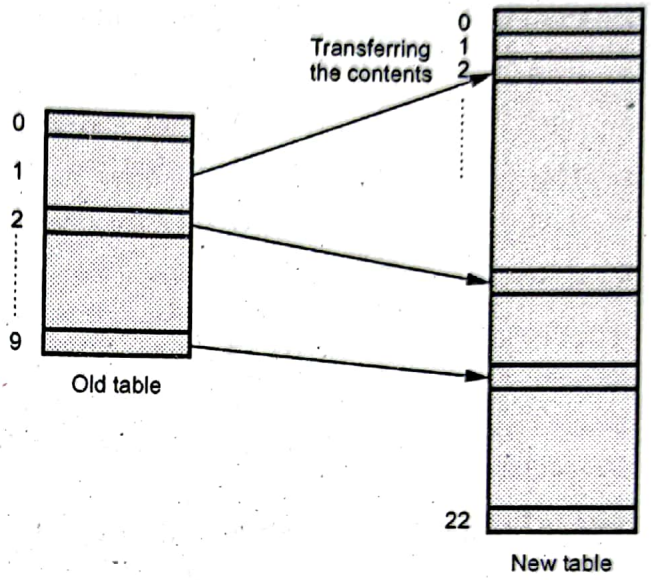


Fig. Q.9.1 Rehashing

- When table is completely full.
- With quadratic probing when the table is filled half.
- When insertions fail due to overflow.
- In such situations, we have to transfer entries from old table to the new table by recomputing their positions using suitable hash functions.

Q.10 What is static and dynamic hashing ? Show that the hash function h(k) = k% 17 does not satisfy the one way property, weak collision resistance or strong collision resistance.

☞[SPPU : May-12, Marks 8]

Ans.: Static hashing :

1. Single hash function h(k) on key k.
2. Desirable properties of a hash fun.

Problems :

1. Fixed size of hash table due to fixed hash function.
2. May require rehasing of all keys when chains or overflow bucket are overfull.

Dynamic hashing :

1. Organize overflow buckets as binary trees.
2. m binary trees for m primary pages.
3. ho(k) produces index of primary page.
4. Particular access structure for binary trees.
5. Family of fun g(k) = {h_1(K) {h_i(K).....}
6. Each h_i(K) produces a bit.

- The function h(K) = k% 17 does not satisfy one way property because it is difficult to find any msg K such h = h(K).
- It doesn't support the weak collision resistance because if given input is m_1 then it is difficult to find another input m_2 such that $m_1 \neq m_2$ and $h(m_1) = h(m_2)$.
- It doesn't support the strong collision resistance because it is difficult to find two different messages m_1 and m_2 such that hash (m_1)= hash(m_2).

Q.11 Create a hash table and resolve collisions using linear probing with and without replacement :
9, 45, 13, 59, 12, 75, 88, 11, 105, 46
Hash table size = 11 and hash function = key mod 10.

☞[SPPU : Dec.-12, Marks 8]

Ans.: i) With chaining without replacement

9, 45, 13, 59, 12,

0		– 1
1		– 1
2	12	– 1
3	13	– 1
4		– 1
5	45	– 1
6		– 1
7		– 1
8		– 1
9	9	10
10	59	– 1

75, 88, 11, 105, 46 0

0	46	– 1
1	11	– 1
2	12	– 1
3	13	– 1
4		
5	45	6
6	75	7
7	105	– 1
8	88	– 1
9	9	10
10	59	– 1

ii) With chaining with replacement

9, 45, 13, 59, 12,

0		
1		
2	12	
3	13	
4		
5	45	
6		
7		
8		
9	9	10
10	59	– 1

75, 88, 11, 105, 46

0	105	– 1
1	11	– 1
2	12	– 1
3	13	– 1
4		
5	45	7
6	46	– 1
7	75	0
8	88	– 1
9	9	10
10	59	– 1

Q.12 Give the input {4371, 1323, 6173, 4199, 4344, 9679, 1989} and hash function h(X) = X(mod 10), show the results for the following :

i) Open addressing hash table using linear probing

ii) Open addressing hash table using quadratic probing

iii) Open addressing hash table with second hash function $h^2(X) = 7 - (X \bmod 7)$.

☞[SPPU : May-08, Marks 12]

Ans.: i) Open addressing hash table using linear probing

- We assume mod function as mod 10.

  4371 mod 10 = 1

  1323 mod 10 = 3

  6173 mod 10 = 3 collision occurs

- Hence by linear probing we will place 6173 at next empty location. That is, at location 4.

  4199 mod 10 = 9

  4344 mod 10 = 4 but location 4 is not empty.

- Hence we will place 4344 at next empty location i.e. 5,

9679 mod 10 = 9 collision occurs so place at next location at 0. The hash table is of size 10. Hence we find the next empty location by rolling the table in forward direction.

1989 mod 10 = 9 collision occurs, so we find the next empty location at index 2.

The hash table will then be

Index	Keys
0	9679
1	4371
2	1989
3	1323
4	6173
5	4344
6	
7	
8	
9	4199

ii) Open addressing hash table using quadratic probing

- In quadratic probing we consider the original hash key and then add an arbitrary polynomial. This sum is then considered for hash function. The hash function will be

$$H(Key) = (Key + i^2) \% m$$

where m can be a table size

If we assume m = 10, then the numbers can be inserted as follows -

Step 1 :

Index	Keys
0	
1	4371
2	
3	
4	
5	
6	
7	
8	
9	

4371 mod 10 = 1

Step 2 :

Index	Keys
0	
1	4371
2	
3	1323
4	
5	
6	
7	
8	
9	

1323 mod 10 = 3

Step 3 :

Index	Keys
0	
1	4371
2	
3	1323
4	6173
5	
6	
7	
8	
9	

6173 mod 10 = 3
As collision occurs
we will apply
quadratic probing.

$\therefore H(key) = (H(key) + i^2) \% m$

Consider i = 0

$\therefore H(6173) = (3 + 0^2) \% 10$

$= 3$ collision

Hence consider i = 1

$H(6173) = (3 + 1^2) \% 10$

$= (3 + 1) \% 10$

$H(6173) = 4$

As index 4 is an empty
slot, we will place
6173 at index 4.

Step 4 :

Index	Keys
0	
1	4371
2	
3	1323
4	6173
5	
6	
7	
8	
9	4199

4199 mod 10 = 9

As this slot is empty
we will place 4199 at index 9

Step 5 :

Index	Keys
0	
1	4371
2	
3	1323
4	6173
5	4344
6	
7	
8	
9	4199

4344 mod 10 = 4
But index 4 shows an occupied slot. Hence collision occurs. Therefore we will place 4344 using quadratic probing.

$\therefore$ H(key) = (H(key) + i^2) % m
H(key) = 4, i = 0, 1, 2, ...
m = 10
$\therefore$ H (key) = (4 + 0^2) % 10 = 4 collision
H (key) = (4 + 1^2) % 10 = 5

The index 5 is an empty slot. Hence we will place 4344 at index 5.

Step 6 :

Index	Key
0	9679
1	4371
2	
3	1323
4	6173
5	4344
6	
7	
8	
9	4199

9679 mod 10 = 9 collision occurs.
Hence we will place the element using quadratic probing.

$\therefore$ H(key) = (H(key) + i^2) % m
H(key) = 9, i will be 0 or 1 or 2 ...
m = 10
H (key) = (9 + 0^2) % 7 $\because$ i = 0
= 9 collision
$\therefore$ H (key) = (9 + 1^2) % 10 = 0
$\therefore$ Place 9679 at index 0

Step 7 :

Index	Key
0	9679
1	4371
2	
3	1323
4	6173
5	4344
6	
7	
8	1989
9	4199

1989 mod 10 = 9 collision occurs.
Hence we will place the element using quadratic probing.

$\therefore$ H(key) = (H(key) + i^2) % m.
H(key) = 9, hence
$\therefore$ H (key) = (9 + 0^2) % 10 $\because$ i = 0
= 9 collision
$\therefore$ H (key) = (9 + 1^2) % 10 $\because$ i = 1
= 0 collision
$\therefore$ H (key) = (9 + 2^2) % 10 $\because$ i = 2
= 3 collison
$\therefore$ H (key) = (9 + 3^2) % 10 $\because$ i = 3
= 8
Insert 1989 at index 8

(iii) Open addressing hash table with second hash function

Step 1 :

Index	Key
0	
1	4371
2	
3	1323
4	
5	
6	
7	
8	
9	

4371 mod 10 = 1

1323 mod 10 = 3

Step 2 :

Index	Key
0	
1	4371
2	
3	1323
4	6173
5	
6	
7	
8	
9	

6173 mod 10 = 3 collision occurs.

Hence we will apply second hash function.

$h2(X) = 7 - (X \bmod 7)$

$h2(6173) = 7 - (6173 \bmod 7)$

$= 7 - 6$

$= 1$

That means we have to take 1 jump from the index 3 (the place at which collision occurs). Hence we will place 6173 at index 4.

Step 3 :

Index	Key
0	
1	4371
2	
3	1323
4	6173
5	
6	
7	
8	
9	4199

4199 mod 10 = 9. As this slot is empty, we will place 4199 at index 9.

Step 4 :

Index	Key
0	
1	4371
2	
3	1323
4	6173
5	9679
6	
7	4344
8	
9	4199

4344 mod 10 = 4 collision occurs.
Hence

$$h(4344) = 7 - (4344 \bmod 7)$$
$$= 7 - 4$$
$$= 3$$

i.e. Take 3 jumps from index 4.

i.e. place 4344 at index 7.

Similarly element 9679 will be placed at index 5 using second hash function.

There is no place for 1989.

Q.13 Assume a hash table of size 10 and Hash function : H(X) = X mod 10
Perform linear probing with and without replacement for the given set of values :
71, 63, 59, 20, 75, 105, 216, 89, 8, 216 ☞[SPPU : May-16, Marks 8]

Ans.:

Chaining without Replacement

0	20	–1
1	71	–1
2	89	4
3	63	–1
4	29	–1
5	75	6
6	105	7
7	216	–1
8	8	–1
9	59	2

Chaining with Replacement

0	20	–1
1	71	–1
2	89	4
3	63	–1
4	29	–1
5	75	7
6	216	–1
7	105	–1
8	8	–1
9	59	2

6.6 : Concept of File

Q.14 What is File ? ☞[SPPU : Dec.-17, June-22, Marks 2]

Ans.: File can be defined as the collection of records in which each record consists of one or more fields.

For example

Roll Number	Name	Marks	Address
10	AAA	90	MG Road
20	BBB	88	Shivaji Nagar

30	CCC	76	S.K.Marg
.	.	.	.
.	.	.	.
.	.	.	.

Fig. Q.14.1 Sample File containing student record

6.7 : File Operations

Q.15 What are different stream classes used for file handling in C++ ?

Ans.: C++ provides following classes to perform input and output of characters to and from the files.

ofstream	This stream class is used to write on files.
ifstream	This stream class is used to read from files.
fstream	This stream class is used for both read and write from/to files.

- To perform file I/O, we need to include **<fstream.h>** in the program. It defines several classes, including **ifstream, ofstream** and **fstream**. These classes are derived from **ios**, so **ifstream, ofstream** and **fstream** have access to all operations defined by **ios**.

Q.16 Explain various file opening modes with respect to text and binary files.

☞[SPPU : May-17, Marks 6]

Ans.:

ios::in	Open for input operation.
ios::out	Open for output operation.
ios::binary	Open for binary operations.
ios::ate	If this flag is set then initial position is set at the end of the file otherwise initial position is at the beginning of the file.
ios::app	The output operations are appended to the file. This is an appending mode. That means contents are inserted at the end of the file.
ios::trunc	The contents of pre existing file get destroyed and it is replaced by new one.

Q.17 Explain 1) seek and 2) tell functions.

Ans.: 1) seek

The seek operation is using two functions seekg and seekp.

- **seekg** means get pointer of specific location for reading of record
- **seekp** means get pointer of specific location for writing of record.

The syntax of **seek** is

seekg (offset, reference - position);

seekp (offset, reference - position);

where, **offset** is any constant specifying the location.

reference - position is for specifying beginning, end or current position. It can be specified as,

ios : : beg	for beginning location
ios : : end	for end of file
ios : : cur	for current location

2) tell

This function tells us the current position.

For example

seqfile. tellg ()	-	gives current position of get pointer (for reading the record).
seqfile. tellp ()	-	gives current position of put pointer (for writing the record).

Q.18 What are primitive operations of sequential file ? Explain with example. **☞[SPPU : May-17, Marks 6]**

Ans.: Primitive operations that can be performed on file are -

1. Creation of file
2. Insertion of records in file
3. Deletion of records from file
4. Modification of records from file
5. Display of records

For example - Refer Q.19 and Q.20.

Q.19 Write C/C++ function for creation of sequential file.

Ans.:

```
void EMP_CLASS::Create()
{
    char ch='y';
    fstream seqfile;
    seqfile.open("EMP.DAT",ios::in|ios::out|ios::binary);
    do
    {
        cout<<"\n Enter Name: ";
        cin>>Records.name;
        cout<<"\n Enter Emp_ID: ";
        cin>>Records.emp_id;
        cout<<"\n Enter Salary: ";
        cin>>Records.salary;
        //then write the record containing this data in the file
        seqfile.write((char*)&Records,sizeof(Records));
        cout<<"\nDo you want to add more records?";
        cin>>ch;
    }while(ch=='y');
    seqfile.close();
}
```

User must enter the desired data in the members of the structure **Records**

Q.20 Write a C/C++ function for display of sequential file.

Ans.:

```
void EMP_CLASS::Display()
{
    fstream seqfile;
    int n,m,i;
    seqfile.open("EMP.DAT",ios::in|ios::out|ios::binary);
    //positioning the pointer in the file at the beginning
    seqfile.seekg(0,ios::beg);
    cout<<"\n The Contents of file are ..."<<endl;
    //read the records sequentially
```

```
    while(seqfile.read((char *)&Records,sizeof(Records)))
    {
        if(Records.emp_id!=-1)
        {
            cout<<"\nName: "<<Records.name;
            cout<<"\nEmp_ID: "<<Records.emp_id;
            cout<<"\nSalary: "<<Records.salary;
            cout<<"\n";
        }
    }
    int last_rec=seqfile.tellg();//last record position
//formula for computing total number of objects in the file
    n=last_rec/(sizeof(Rec));
    cout<<"\n\n Total number of objects are "<<n<<"(considering
logical deletion)";
    seqfile.close();
}
```

Q.21 Distinguish between logical and physical deletion of records and illustrate it with example.

☞[SPPU : May-15, Marks 6, Dec.-15, Marks 4]

Ans.: The record can be deleted logically or physically.

In case of physical deletion, the records are physically removed or actually removed from the file. Such a record will not be available or can not be restored after deletion. By physical deletion memory occupied by that record is free for use. In case of logical deletion, the record is simply marked as deleted by some specific value. In this case record is physically present but it is always ignored as if it is deleted. We can restore such record if required. The drawback of this type of deletion is that memory remain occupied by this record.

Q.22 Write a Pseudo code for search and insert operations in index sequential file. ☞[SPPU : Dec.-17, Marks 6]

Ans.: (1) Insert function

```
void EMP_CLASS::Create()
{
    int i,j;
    char ch='y';
    fstream seqfile;
    fstream indexfile;
    i=0;
    indexfile.open("IND.DAT",ios::in|ios::out|ios::binary);
    seqfile.open("EMP.DAT",ios::in|ios::out|ios::binary);
    do
    {
        cout<<"\n Enter Name: ";
        cin>>Records.name;
        cout<<"\n Enter Emp_ID: ";
        cin>>Records.emp_id;
        cout<<"\n Enter Salary: ";
        cin>>Records.salary;

        seqfile.write((char*)&Records,
        sizeof(Records))<<flush;

        Ind_Records.emp_id=Records.emp_id;
        Ind_Records.position=i;
        indexfile.write((char*)&Ind_Records,sizeof(Ind_Records))
        <<flush;
        i++;
        cout<<"\nDo you want to add more records?";
        cin>>ch;
    }while(ch=='y');
    seqfile.close();
    indexfile.close();
}
```

Opening both the files for reading and writing purpose

User must enter some records to store them first in the sequential file.

Records for index file are created from seq.file and writing them to ind.file

(2) Search function

```
void EMP_CLASS::Search()
{
   fstream seqfile;
   fstream indexfile;
   int id,pos,offset;
   cout<<"\n Enter the Emp_ID for searching the record ";
   cin>>id;
   indexfile.open("IND.DAT",ios::in|ios::binary);
   pos=-1;
   //reading index file to obtain the index of desired record
   while(indexfile.read((char *)&Ind_Records,sizeof(Ind_Records)))
   {
      if(id==Ind_Records.emp_id)//desired record found
      {
         pos=Ind_Records.position;//seeking the position
         break;
      }
   }
   if(pos==-1)
   {
      cout<<"\n Record is not present in the file";
      return;
   }
//calculate offset using position obtained from ind. file
   offset=pos*sizeof(Records);
   seqfile.open("EMP.DAT",ios::in|ios::binary);
//seeking the record from seq. file using calculated offset
   seqfile.seekg(offset,ios::beg);//seeking for reading purpose
   seqfile.read((char *)&Records,sizeof(Records));
   if(Records.emp_id==-1)
   {
      cout<<"\n Record is not present in the file";
      return;
   }
```

```
    else //emp_id=desired record's id
    {
       cout<<"\n The Record is present in the file and it is...";
       cout<<"\n Name: "<<Records.name;
       cout<<"\n Emp_ID: "<<Records.emp_id;
       cout<<"\n Salary: "<<Records.salary;
    }
    seqfile.close();
    indexfile.close();
}
```

Q.23 With C++ program to perform the following operations on direct access file.
(i) Create and display records (ii) Insert record

☞[SPPU : May-17, Marks 6]

Ans.:

```
void EMP_CLASS::Insert()
{
   int i,h;
   char ch='y';
   char new_name[10];
   int new_emp_id;
   int new_salary;
   fstream seqfile;
   init();//initialising the hash table
   seqfile.open("EMP.DAT",ios::in|ios::out|ios::binary);
   do
   {
      cout<<"\n Enter Name: ";
      cin>>new_name;
      cout<<"\n Enter Emp_ID: ";
```

```
cin>>new_emp_id;
cout<<"\n Enter Salary: ";
cin>>new_salary;
h=Hash(new_emp_id);
int offset;
offset=h*sizeof(Records);

//seeking for reading record
seqfile.seekg(offset);
seqfile.read((char*)&Records,sizeof(Records));
//seeking for writing record
seqfile.seekp(offset);
if(Records.emp_id==-1)
{
 strcpy(Records.name,new_name);
 Records.emp_id=new_emp_id;
 Records.salary=new_salary;
 Records.link=-1;
 Records.loc=h;//h is used for marking the loc.
 seqfile.write((char*)&Records,sizeof(Records))<<flush;
//thus rec. is inserted at the hashed position in file
}
else//collision occurs
{
int flag=0;
int prev_link=Records.loc;
do    //handling collision
{
h++;//searching down for empty loc.in the file
if(h>size+1)
{
  cout<<"\n The hash table is Full, Can't insert record!!!";
  return;
```

Calling Hash function to get the direct location for the record in the file. Function returns a hash key in variable **h**

Using **h** for getting actual position in the file. Hence offset is calculated

```
        }
        offset=h*sizeof(Records);
        seqfile.seekg(offset);
        seqfile.read((char*)&Records,sizeof(Records));
        if(Records.emp_id==-1) //finding empty loc. using linear
                               // probing
        {
            seqfile.seekp(offset);//seeking the empty slot in the file
            strcpy(Records.name,new_name);//for placing the record
            Records.emp_id=new_emp_id;
            Records.salary=new_salary;
            Records.link=-1;
            Records.loc=h;//setting the location for colliding record
            seqfile.write((char*)&Records,sizeof(Records))<<flush;
           //collinding record is placed in the file at proper pos.
       //chain table is maintained for keeping track of all the
       // colliding entries.
            Chain_tab[prev_link][h]=1;
            flag=1;//indicates colliding record is inserted
        }//end of if
       }while(flag==0);//collision handled
       }//end of else
       cout<<"\nDo you want to add more records?";
       cin>>ch;
       chain_wo_repl();//setting the chain to handle collision
   }while(ch=='y');
   seqfile.close();
}
void EMP_CLASS::Display()
{
   fstream seqfile;
   seqfile.open("EMP.DAT",ios::in|ios::out|ios::binary);
   seqfile.seekg(0,ios::beg);
   cout<<"\n The Contents of file are ..."<<endl;
   cout<<"\nLoc.    Name      Emp_ID    Salary      Link  ";
```

```
    while(seqfile.read((char *)&Records,sizeof(Records)))
    {

          if(strcmp(Records.name,"")!=0)//not displaying empty slots
          {
          cout<<"\n----------------------------------------------------\n";
    cout<<Records.loc<<"     "<<Records.name<<flush<<"
"<<Records.emp_id;
    cout<<"          "<<Records.salary<<"          "<<Records.link;

           }
    }
 seqfile.close();
}
void EMP_CLASS::chain_wo_repl()
{
fstream seqfile;
int i,j,h,offset;
seqfile.open("EMP.DAT",ios::in|ios::out|ios::binary);
for(i=0;i<size;i++)
  {
     h=i;
     for(j=0;j<size;j++)
     {
        if(Chain_tab[i][j]==1)
        {
         offset=h*sizeof(Records);
         seqfile.seekg(offset);
         seqfile.read((char*)&Records,sizeof(Records));
         seqfile.seekp(offset);
         Records.link=j;
         seqfile.write((char*)&Records,sizeof(Records));
```

```
            h=j;
          }
        }
      }
    seqfile.close();
}
```

6.8 : File Types and File Organization

Q.24 Explain different types of file organizations.

☞[SPPU : Dec.-17, June-22, Marks 4]

Ans.: Definition : File organization deals with representation or arrangement of data on external storage.

- **Access Methods**

The method by which records can be retrieved or updated from the file is called access method. Records within the file can be organized in variety of ways. Fig. Q.24.1 shows various types by which a file can be organized -

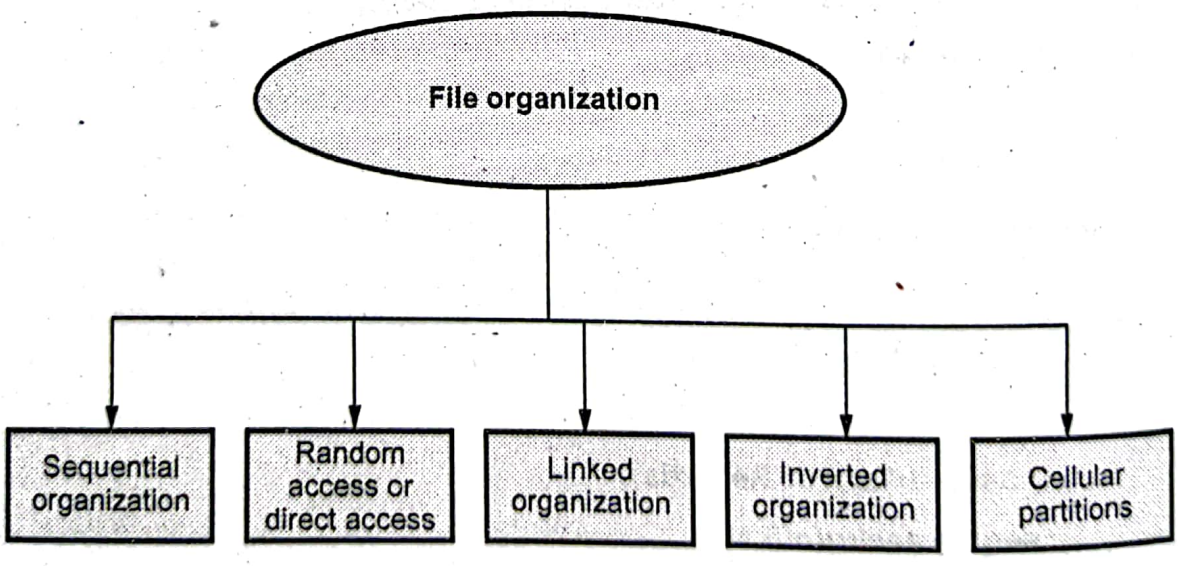


Fig. Q.24.1 Types of file organizations

Q.25 Compare Binary File and Text File.

☞[SPPU : Dec.-17, Marks 6]

Ans.:

Sr. No.	Text file	Binary file
1.	Text files contain plain text data.	The binary file contain the data in binary form.
2.	The text file does not contain the graphical data.	The binary file can store the data such as text, graphics, image and sound.
3.	Text files can be directly read and interpreted.	The binary files cannot be read directly. With the help of some tool the binary file can be read. For example HexDump is used to read the binary files.
4.	The text files are not executable files.	The binary files can be executable files.
5.	Consume more memory.	Consume less memory.
6.	Accessing time is more as compared to binary file.	Accessing time is less as compared to text file.

Q.26 Explain sequential file organization.

☞ **[SPPU : June-22, Marks 2]**

Ans. : • This type of file organization makes use of ISMA scheme.

- The ISMA scheme i.e. Index Sequential Memory Access scheme is a kind of file organization in which record can be arranged using cylinder and surfaces.

For example : Consider the arrangement of students' record in which cylinders and surfaces are used.

Record number	Name	Roll_number	Marks	Cylinder	Surface
1	AAA	10	70	1	1
2	BBB	20	90	1	2
3	CCC	30	80	2	1

4	DDD	40	55	2	2
5	EEE	50	78	3	1

Now if we want to find record of Roll_number 40 then we can search it based on the cylinder. That is we will first search for the cylinder number 2, then we will make the search based on the surface. When we reach at the surface 2 of cylinder 2 then the desired record can be obtained within the track.

Q.27 Explain direct access file organization.

☞ **[SPPU : June-22, Marks 2]**

Ans. : Random organization is a kind of file organization in which records are stored at random locations on the disks.

There are three techniques used in random organization

Let us discuss each of them -

1. Direct addressing : In direct addressing two types of records are handled : fixed length record and variable length record.

- For storing the fixed length records the disk space is divided into the nodes. These nodes are large enough to hold individual record.
- Every fixed length record is stored in node number # which is equal to the primary key value. For example: If a primary key value is 185 then the record must be present in the node number 185.

For example :

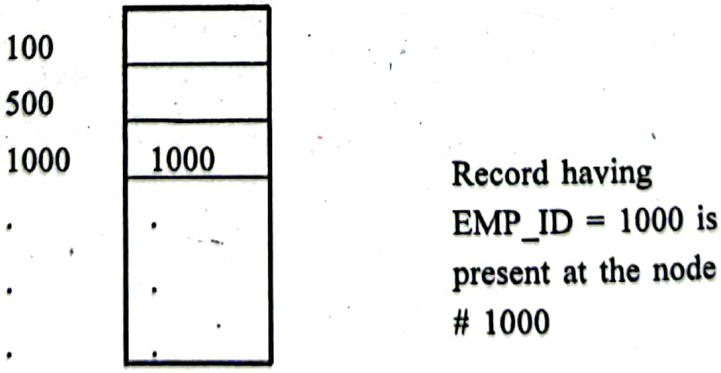


Fig. Q.27.1 Storing of fixed length record

- For storing the **variable length records** on the disk, the address (pointer) of each individual record is stored in the file at specific index. We can locate the variable length record using the index of the pointer. This pointer will point to desired record which is present on the disk.

2. Directory lookup : • In this scheme the index for the pointers to the records is maintained.

- For retrieving the desired record first of all the index for the record address is searched and then using this record address the actual record is accessed.

3. Hashing : • Hashing is a technique in which hash key is obtained using some suitable **hash function** and record is placed in the hash table with the help of this **hash key.**

Thus in this random organization, the record can be quickly searched with the help of hash function being used.

Q.28 Explain the concept of index sequential file organization.

[SPPU : June-22, Marks 2]

Ans. : • The main drawback of sequential file is that searching operation is not efficient. Because in sequential organization primary key of every record is compared with searching key. To optimize this operation concept of index sequential file is introduced.

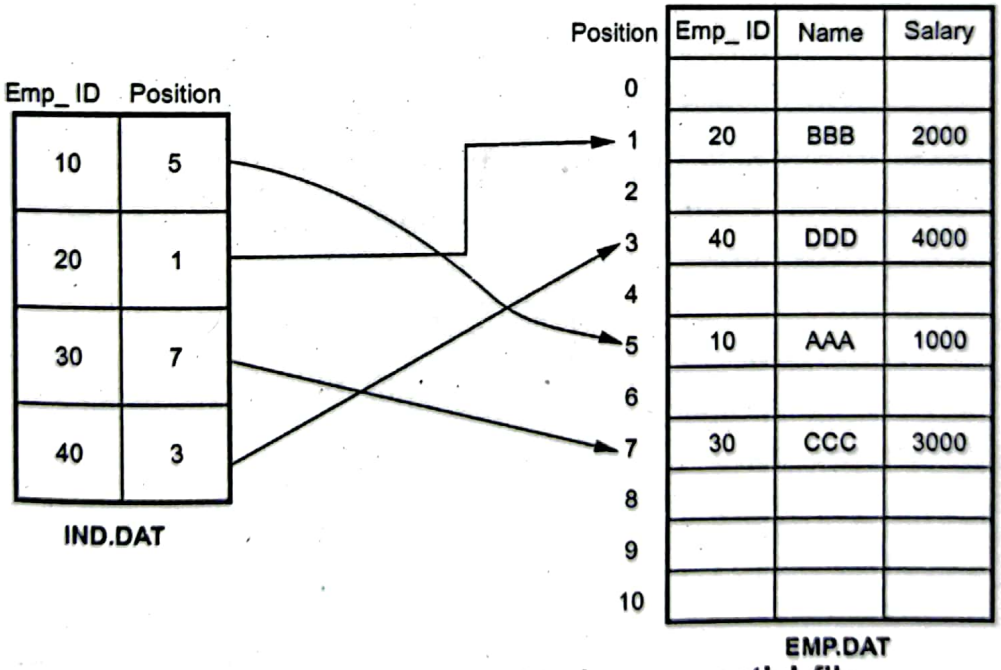


Fig. Q.28.1 Structure of index sequential file

- In index sequential file organization, a separate file for storing indexes of every record is maintained along with the master file.

For example

There are two files "IND.DAT" and "EMP.DAT".

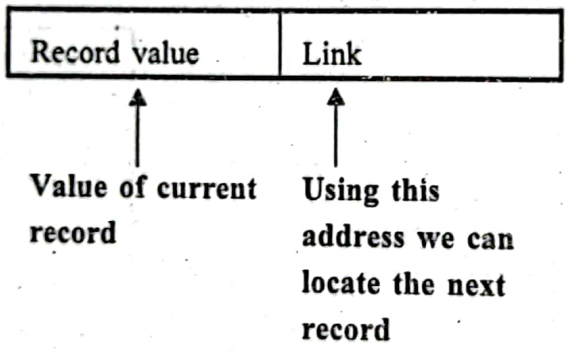
Using the positions field of "IND.DAT" file we can quickly retrieved the desired record.

- The index sequential organization, accelerates the retrieval of any desired record. In this case, we need not have to scan the entire memory block of record. Instead of that using primary key (such as EMP_ID) and position we can access the record from master file.

Q.29 Write short note on linked organization.

Ans. : • In linked organization the logical sequence of the records is different than the physical sequence. In any sequential organization if we are accessing n^{th} node at Loc_i then $(n+1)^{th}$ record may be located at (Loc_i+c) where c is the constant which represents the length of the record or it may be some inter-record spacing.

- In linked organization we can access next logical record by following the **link-value** pair. The link-value pair denotes each individual record.
- The typical structure of every record is as follows -



Thus records in the linked organization can be stored as follows -

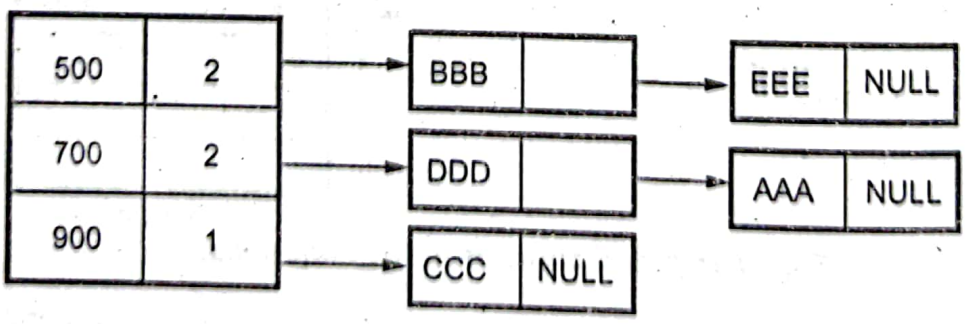
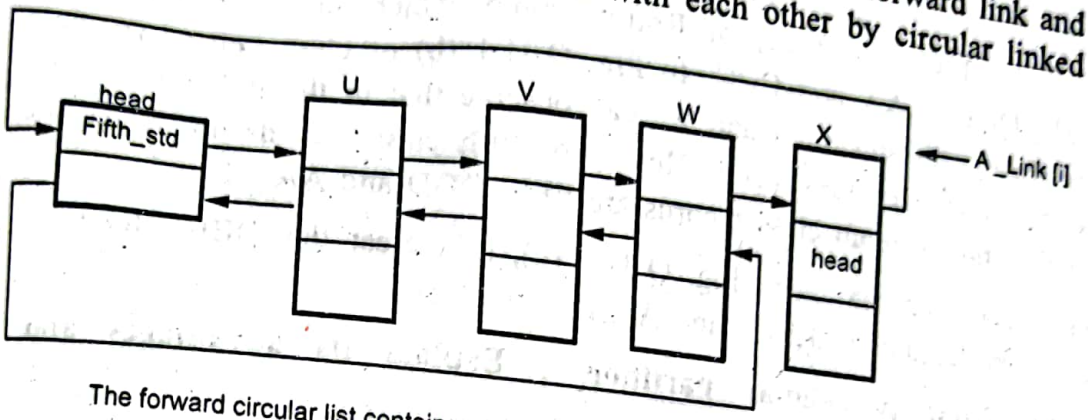


Fig. Q.29.1 Roll_No Index

Q.30 Explain the concept of coral ring.

Ans. : Coral ring is a file organization in which doubly linked multi-list structure is maintained. Each list is a circular list. Thus coral ring structure is a kind of structure in which **circular doubly** linked list is maintained for connecting the records together.

Associated with each record there are two link fields i.e. forward link and back link. Thus records get associated with each other by circular linked list.



The forward circular list contains nodes head U, V, W, X.

The reverse circular list contains nodes head W, V, U.

Fig. Q.30.1 Coral ring structure

Q.31 What is inverted file structure? Explain with suitable example

Ans. : • Inverted files are similar to multi-lists.

• The difference between multi-list and inverted files is that in multi-lists records with the same key value are linked together along with the **link information** being kept in **individual record.** But in case of inverted files this **link information** is kept in the **index itself.**

For example

437	BBB
488	EEE
689	DDD
695	AAA
778	CCC

Fig. Q.31.1 (a) Roll_No Index

Fifth	BBB, CCC
Tenth	EEE, DDD, AAA

Fig. Q.31.1 (b) Class index

Female	BBB, DDD, CCC
Male	EEE, AAA

Fig. Q.31.1 (c) Sex Index

First class	EEE
Second class	AAA, DDD, CCC
Pass class	BBB

Fig. Q.31.1 (d) Marks Index

Consider Fig. Q.31.1 (a) of Roll-no index which shows records BBB, EEE, DDD, AAA and CCC. In Fig. Q.31.1 (b) of class index two class are there fifth and tenth and we can observe that in the link information is stored in the index itself. Hence for fifth class records are BBB and CCC. And for tenth class records are EEE, DDD and AAA .

Similarly from sex index Fig. Q.31.1 (c), it is clear that BBB, DDD and CCC are females and EEE and AAA are males.

Q.32 What is cellular partition ? Explain its advantages and disadvantages.

Ans. : • For reducing the searching time during file operations, the storage media (e.g. secondary memory, magnetic disk, magnetic tape etc.) may be divided into cells.

- The cells can be of two types -
  i) Entire disk pack can be a cell
  ii) A cylinder can be a cell.
- A list of records can occupy either entire disk pack or it may lie on particular cylinder.
- If all the records lie on the same cylinder then without moving the read/write head the records can be accessed.
- If the cell is nothing but entire disk pack then the disk is partitioned into different partitions. Such partitions are called **cellular partitions.** Then these different cells can be searched in parallel.

Advantages of cellular partitions

1) Various read operations can be performed parallely in order to reduce the search time.
2) Faster execution of any query.

Disadvantage

1) If multiple records lie in the same cell then reading a single cell becomes a time consuming process.

6.9 : Comparison of Different File Organizations

Q.33 Compare sequential and direct access files.

Ans. : ☞[SPPU : June-22, Marks 2]

Sr. No.	Sequential file	Direct access file
1.	The records are inserted in the file sequentially as they get read.	The records are inserted at some specific locations in the file. These locations are obtained using hash function.
2.	No collision handling required.	Collision handling techniques are required.
3.	Slower access method and records can be retrieved by searching the file sequentially.	Faster access method and records can be retrieved directly with the help of hash key.
4.	Simple to implement.	Complex to implement because computation of hash function and collision handling is required.

Q.34 Compare direct access file and index sequential file.

☞[SPPU : June-22, Marks 2]

Ans. :

Sr. No.	Direct access file	Index sequential file
1.	Desired record can be obtained using the hash key. This hash key is returned by some hash function.	Desired record can be obtained using the index which is maintained in a separate file.

2.	On insertion or deletion of records, collision may occur. Hence collision handling techniques are required.	Insertion and deletion operations can be performed by simple index manipulation.
3.	Record length must be fixed.	Variable length records are allowed.
4.	Hash key is obtained by passing primary key of record to hash function.	Index key is obtained using primary key of the record.
5.	Records can be arranged randomly. This arrangement is influenced by hash key.	Records are arranged sequentially in the master file.

END...