Chapter No-2

Long

4. A Linear search looks down a list, one item at a time, without jumping. In Complexity terms this an O(n) search. The time taken to search the list gets bigger that binary search.

In Binary search when we start with the middle of a sorted list, and whether that's greater than or less than the value we are looking for, which determines whether the value is in the first or second half of the list. Jump to the half way through the sub list and compare again etc.

Chapter No-3

Short –

4. Run time of worst case analysis for quick sort is $0(n^2)$

5. Worst case for quick sort is in following situation-

(i) When the array is already sorted

(ii) When the array is already in descending order

(iii) When all elements are equal

Long –

2. Even though quick sort has a worst case run time $0(n^2)$, quick sort is considered the best sorting because it is very efficient on the average, its expected running time is 0(n log n) where the constant are very small compared to other sorting algorithm.

Chapter No-5

Short –

3. Linked lists have several advantages over arrays. Elements can be inserted into linked lists indefinitely, while an array will eventually either fill up or need to be resized,

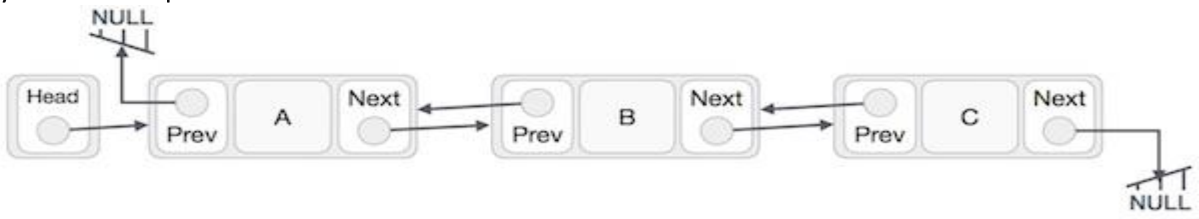
On the other hand, arrays allow random access, while linked lists allow only sequential access to elements. Singly-linked lists, in fact, can only be traversed in one direction. This makes linked lists unsuitable for applications where it's useful to look up an element by its index quickly, such as heapsort. Sequential access on arrays is also faster than on linked lists on many machines due to locality of reference and data caches. Linked lists receive almost no benefit from the cache.

Another disadvantage of linked lists is the extra storage needed for references, which often makes them impractical for lists of small data items such as characters or boolean values

Long

1.

Doubly Linked List Representation



- Doubly Linked List contains a link element called first and last.
- Each link carries a data field(s) and two link fields called next and prev.
- Each link is linked with its next link using its next link.
- Each link is linked with its previous link using its previous link.
- The last link carries a link as null to mark the end of the list.

2. Difference between Singly and Doubly linked list

1 Singly linked list allows you to go one way direction
Doubly linked list has two way directions next and previous

2 Singly linked list uses less memory per node (one pointer)

Doubly linked list uses More memory per node than Singly Linked list (two pointers)

3 There is a little-known trick that lets you delete from a singly-linked list in O(1)
Doubly-linked lists can be used in places where singly-linked lists would not work (a doubly-ended queue),

4 Complexity of Insertion and Deletion at known position is O (n).
Complexity of Insertion and Deletion at known position is O (1).

5 If we need to save memory in need to update node values frequently and searching is not required, we
can use Singly Linked list.
If we need faster performance in searching and memory is not a limitation we use Doubly Linked List

8 In single list Each node contains at least two parts:
a) info
b) link
In doubly linked list Each node contains at least three parts:
a) info
b) link to next node
c) link to previous node

Chapter No-9

Short

2. Static data members

- A data member of a class can be qualified as static.
- A static member variable has certain special characteristics.
  They are:
  - It is initialized to zero when the first object of its class is created. No other initialization is permitted.
  - Only one copy of that member is created for the entire class and is shared by all the objects of that class, no matter how many objects are created.
  - It is visible only within the class, but its lifetime is the entire program.
  - Static variables are normally used to maintain values common to the entire class.
  - The type and the scope of each static member variable must be defined outside the class definition.
  - They are also known as class variables.

3. Static Member Function

1. A static member function can only have access to other static data members and functions declared in the same class.
2. A static member function can be called using the class name with a scope resolution operator instead of object name.
3. Global functions and data may be accessed by static member function.
4. A static member function does not have a this Pointer.
5. There can not be a static and a non-static version of the same function.
6. They can not be declared as const or volatile.
7. A static member function may not be virtual.

Long

3.

```
#include <iostream.h>
#include <conio.h>
class b;
class a
{
int x;
public:
void assign(int t)
{
```

```
x =t;
}
void display()
{
cout<<"value of x is: "<<x<<endl;
}
friend void swap(a&,b&);
};
class b
{
int y;
public:
void assign(int w)
{
y=w;
}
void display()
{
cout<<"value of y is: "<<y<<endl;
}
friend void swap(a&,b&);
};
void swap(a &c,b&d)
{
int temp;
temp =c.x;
c.x = d.y;
d.y = temp;
}
void main()
{
a n;
b m;
int a,b;
cout<<"enter x =";
cin>>a ;
cout<<"Enter y =";
cin>>b;
n.assign(a);
m.assign(b);
swap(n,m);
n.display();
m.display();
getch();
}
```

Chapter No-11

Long

2.

```
#include<iostream.h>
#include<string.h>
#include<conio.h>
class String
```

```
{
	public:
		char str[20];
	public:
		void accept_string()
		{
			cout<<"\n Enter String		:   ";
			cin>>str;
		}
		void display_string()
		{
			cout<<str;
		}
		String operator+(String x)  //Concatenating String
		{
			String s;
			strcat(str,x.str);
			strcpy(s.str,str);
			return s;
		}
};
void main()
{
	clrscr();
	String str1, str2, str3;
	    str1.accept_string();
	    str2.accept_string();
	    str3=str1+str2;	//String is concatenated. Overloaded '+' operator
	    str3.display_string();
	    getch();
}
```

Chapter No-13

Very Short

4.

Database Schema

A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data.

A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database.

5.

Index in sql is created on existing tables to retrieve the rows quickly.

When there are thousands of records in a table, retrieving information will take a long time. Therefore indexes are created on columns which are accessed frequently, so that the information can be retrieved quickly. Indexes can be created on a single column or a group of columns.

The CREATE INDEX Command

The basic syntax of a **CREATE INDEX** is as follows.

```
CREATE INDEX index_name ON table_name;
```

Single-Column Indexes

A single-column index is created based on only one table column. The basic syntax is as follows.

```
CREATE INDEX index_name
ON table_name (column_name);
```

Unique Indexes

Unique indexes are used not only for performance, but also for data integrity. A unique index does not allow any duplicate values to be inserted into the table. The basic syntax is as follows.

```
CREATE UNIQUE INDEX index_name
on table_name (column_name);
```

Composite Indexes

A composite index is an index on two or more columns of a table. Its basic syntax is as follows.

```
CREATE INDEX index_name
on table_name (column1, column2);
```

Whether to create a single-column index or a composite index, take into consideration the column(s) that you may use very frequently in a query's WHERE clause as filter conditions.
Should there be only one column used, a single-column index should be the choice. Should there be two or more columns that are frequently used in the WHERE clause as filters, the composite index would be the best choice.

Implicit Indexes

Implicit indexes are indexes that are automatically created by the database server when an object is created. Indexes are automatically created for primary key constraints and unique constraints.

The DROP INDEX Command

An index can be dropped using SQL **DROP** command. Care should be taken when dropping an index because the performance may either slow down or improve.
The basic syntax is as follows −

```
DROP INDEX index_name;
```

You can check the INDEX Constraint chapter to see some actual examples on Indexes.

8.

Definition of schema: Design of a database is called the schema. Schema is of three types: Physical schema, logical schema and view schema.

Definition of instance: The data stored in database at a particular moment of time is called instance of database. Database schema defines the variable declarations in tables that belong to a particular database; the value of these variables at a moment of time is called the instance of that database.

9. **Step 1:** Define the Purpose of the Database (Requirement Analysis) Gather the requirements and define the objective of your database
Step 2: Gather Data, Organize in tables and Specify the Primary Keys.
Step 3: Create Relationships among Tables.
Step 4: Refine & Normalize the Design.

Short

1. Atomicity − This property states that a transaction must be treated as an atomic unit, that is, either all of its operations are executed or none. There must be no state in a database where a transaction is left partially completed.

5. Bad Database

A badly designed database has the following problems:

1. Related data is scattered over various tables. A change must be updated at many places
2. Data is inconsistent or ambiguous
3. The database is unnecessary complex, hacked with lots of tricks.
4. The database has 'hidden' information

5. The database is slow, inflexible, hard to extend and can not handle all real life situations.

Chapter No-14

Short

5. The CARTESIAN JOIN or CROSS JOIN returns the Cartesian product of the sets of records from two or more joined tables. Thus, it equates to an inner join where the join-condition always evaluates to either True or where the join-condition is absent from the statement.

Syntax

The basic syntax of the **CARTESIAN JOIN** or the **CROSS JOIN** is as follows –

```
SELECT table1.column1, table2.column2... FROM  table1, table2 [, table3 ]
```