* Structs: **5**

Binding data into a group, like a blue print to store data and perform actions in same way just like a Human Being – Everybody has nose for breathing, here nose is data and breathing is a function.

* Classes: **5**

Binding of data with some access specifiers which means the data and functionality is only accessible by selected members of that class. Example – Golf club, here anyone can join the club and play but can’t access and take benefits if not the member. (Only Private)

* Encapsulation: **5**

Concept of binding data which is used in Classes and Structures.

* Composition: **5**

It’s like adding a functionality which has multiuse. For example – Rotate Wheel functionality for Cars and Bikes, here rotation function is reusable and can work with both structures.

* Pointers: **5**

Helps with dynamic programming and Data Structures.

* Pass-by-reference: **5**

Instead of passing a copy of any instance pass by reference passes the address so the execution time could be faster and the operation would be happening on the actual data.

* Linked lists: **5**

List of nodes connected with the pointers to the next node. Helps creating and adding more data in the system, substitute of Array Data Structure. Example apps: Tinder – Swiping, here when user swipe it either save into the memory or delete from the list according to the action and functionality.

* Binary search trees: **5**

Ordering data into a tree structure which make searching easy and faster. It can be implemented by arrays or LinkedIn either.

* File input and output: **4**

Operations on the files with input streams and output streams. Saving, Deleting, Reading, Writing data into the file.

* Private member methods: **5**

Private members which can’t be accessible by another instance or function but own classes. This gives a security of overwriting of data from the outside of that encapsulated structure.