OOP

1. What is encapsulation? The attribute of an object to hide the data and set access specifiers for it is known as encapsulation. Access specifers are of three types which are as follows: 1. Public 2. Protected 3. Private

2. What is polymorphism? The attribute to exist in multiple forms is known as polymorphism. as per object attribute it means having multiple definitions of a method. Types: 1. Static (occurs at compile time i.e. method overloading) 2. Dynamic (occurs at runtime i.e. method overriding)

3. What is method overloadding and overriding? Method overloading: Creating metods of same name but with diffrent arguments.

   Method overriding A feature to redefine the base class methods as per the child class need but the signature meaning, argument passed and return type will remain same.

4. What is inheritance? The concept where one class can share its structure and behavior to another class is called inheritance. there are several types of inheritance which are as follows: 1. Single inheritance 2. Multiple inheritance 3. Multilevel inheritance 4. Hybrid inheritance

5. What is data abstraction? Hiding of implementation details and displaying only relevant information is known as data abstraction. Data abstraction techniques: 1. Abstract Class 2. Abstract Methods

   Abstract class consist of abstract methods and these methods are declared but not defined if these methods are to be used in any subsclass then these methods needs to be exclusively defined in that subclass

6. virtual functions? These are defined in the parent class with virtual keywords and are overriden in the subclass these are used to achieve runtime polymorphism.

7. What are access specifiers and why they're used? access modifiers / specifiers are basically attributes of a class which determines the access scope of its methods or variables. Types of access specifiers are as follows: 1. Public 2. Default 3. Private 4. Protected 5. Friend 6. Protected Friend

   Level of access:

```
 |    Conditions   | Public  | Default | Private  | Protected | Friend  |
Protected Firend |
 |  -------------- | ------- | ------- | -------- | --------- | ------- | --
-------------- |
 |  Class          | - [x]   | - [x]   |  - [x]   |   - [x]   |  - [x]  |
- [x]       |
 |  subclass       | - [x]   | - [x]   |  - [ ]   |   - [ ]   |  - [x]  |
- [ ]       |
 |  Derived Class  | - [x]   | - [x]   |  - [ ]   |   - [x]   |  - [x]  |
- [ ]       |
 |  Globally       | - [x]   | - [ ]   |  - [ ]   |   - [ ]   |  - [ ]  |
- [ ]       |
```

```
 |  Same Package   | - [x]   | - [ ]   |  - [ ]   |   - [ ]   |  - [x]  |
- [x]       |
 |  Diff. Package  | - [x]   | - [ ]   |  - [ ]   |   - [ ]   |  - [ ]  |
- [x]       |
```

8. How do we modify protected data? By using setter and getters

9. Why setters, getters are kept public but the data is kept private? Apart from data encapsulation and easier future modifications here is a list of pros for keeping them public: 1. Encapsulation of behavior associated with getting or setting the property - this allows additional functionality (like validation) to be added more easily later. 2. Hiding the internal representation of the property while exposing a property using an alternative representation. 3. Insulating your public interface from change - allowing the public interface to remain constant while the implementation changes without affecting existing consumers. 4. Controlling the lifetime and memory management (disposal) semantics of the property - particularly important in non-managed memory environments (like C++ or Objective-C). 5. Providing a debugging interception point for when a property changes at runtime - debugging when and where a property changed to a particular value can be quite difficult without this in some languages. 6. Improved interoperability with libraries that are designed to operate against property getter/setters - Mocking, Serialization, and WPF come to mind. 7. Allowing inheritors to change the semantics of how the property behaves and is exposed by overriding the getter/setter methods. 8. Allowing the getter/setter to be passed around as lambda expressions rather than values. 9. Getters and setters can allow different access levels - for example the get may be public, but the set could be protected.

10. What are different types of argument? A parameter is a variable used during the declaration of the function or subroutine, and arguments are passed to the function body, and it should match with the parameter defined. There are two types of Arguments.

    ```
    1. Call by Value – Value passed will get modified only inside the function,
    and it returns the same value whatever it is passed into the function.
    2. Call by Reference – Value passed will get modified in both inside and
    outside the functions and it returns the same or different value.
    ```

11. What is an interface? In simple words each class can have its own interface and an interface is like a specsheet of attributes which is used to determine whether the instance qualify to be known as an object or not.

    or

    An interface is a collection of an abstract method. If the class implements an interface, it thereby inherits all the abstract methods of an interface.

    To create an interface, interface keyword is used.

12. What is exception handling? An exception is an event that occurs during the execution of a program. Exceptions can be of any type – Runtime exception, Error exceptions. Those exceptions are adequately handled through exception handling mechanism like try, catch, and throw keywords.

13. What is a friend function? A friend function is a friend of a class that is allowed to access to Public, private, or protected data in that same class. If the function is defined outside the class cannot access such information.

    A friend can be declared anywhere in the class declaration, and it cannot be affected by access control keywords like private, public, or protected.

14. What is this pointer? THIS pointer refers to the current object of a class. THIS keyword is used as a pointer which differentiates between the current object with the global object. It refers to the current object.

15. What is difference between structure and a class? Class: User-defined blueprint from which objects are created. It consists of methods or set of instructions that are to be performed on the objects.

    Structure: A structure is basically a user-defined collection of variables which are of different data types.

16. What is pure virtual function? Pure virtual functions or abstract functions are functions that are only declared in the base class. This means that they do not contain any definition in the base class and need to be redefined in the subclass.

17. How is overloading and overriding done? Overloading is static Binding, whereas Overriding is dynamic Binding. Overloading is nothing but the same method with different arguments, and it may or may not return the equal value in the same class itself. Operator keyword is used for overloading.

    Overriding is the same method names with the same arguments and return types associated with the class and its child class.

18. What is the difference between OOP and SOP?

```
| Object oriented programming                                          |
Structural Programming
|
| ---------------------------------------------------------------    | --
-------------------------------------------------------------------    |
|  OPP is  based on objects rather than just functions and procedures   |
Provides logical structure to a program where programs are divided functions
|
|  Bottom-up approach                                                  |
Top-down approach
|
|  Provides data hiding                                                |
Does not provide data hiding
|
|  Can solve problems of any complexity                                |
Can solve moderate problems
|
|  Code can be reused thereby reducing redundancy                      |
Does not support code reusability
|
```

19. What is Data abtraction and its techniques? Data abstraction is a very important feature of OOPs that allows displaying only the important information and hiding the implementation details. For example, while riding a bike, you know that if you raise the accelerator, the speed will increase, but you don't know how it actually happens. This is data abstraction as the implementation details are hidden from the rider. Data abstraction can be achieved through:

    ```
    Abstract class
        An abstract class is a class that consists of abstract methods. These
    methods are basically declared but not defined. If these methods are to be
    used in some subclass, they need to be exclusively defined in the subclass.

    Abstract method
    ```

20. What is finalize keyword? Finalize method helps to perform cleanup operations on the resources which are not currently used. Finalize method is protected, and it is accessible only through this class or by a derived class. Finalize as an object method used to free up unmanned resources and cleanup before Garbage Collection(GC). It performs memory management tasks.

21. What is Garbage Collection (GC)? GC is an implementation of automatic memory management. The Garbage collector frees up space occupied by objects that are no longer in existence.

22. What is a final variable? A variable whose value does not change. It always refers to the same object by the property of non-transversity.

Data Structure & Algorithm

Time Complexities

Data Structures

```
        +--------------------+-------------------------------------------------
-----------------------------------------+------------------+
        |   Data Structure   |                                         Time
Complexity                                      | Space Complexity |
        +--------------------+-------------------------------------------------+----
-----------------------------------------+------------------+
        |                    |                   Average                     |
Worst                     |       Worst      |
        +--------------------+-----------+-----------+-----------+-----------+----
-------+-----------+-----------+-----------+------------------+
        |                    |   Access  |   Search  | Insertion |  Deletion |
Access  |   Search  | Insertion |  Deletion |                  |
        +--------------------+-----------+-----------+-----------+-----------+----
-------+-----------+-----------+-----------+------------------+
        |        Array       |    0(1)   |    0(n)   |    0(n)   |    0(n)   |
O(1)   |    O(n)   |    O(n)   |    O(n)   |       O(n)       |
```

```
        +--------------------+-----------+-----------+-----------+-----------+----
-------+-----------+-----------+-----------+------------------+
        |        Stack       |    0(n)   |    0(n)   |    0(1)   |    0(1)   |
O(n)   |    O(n)   |    O(1)   |    O(1)   |       O(n)       |
        +--------------------+-----------+-----------+-----------+-----------+----
-------+-----------+-----------+-----------+------------------+
        |        Queue       |    0(n)   |    0(n)   |    0(1)   |    0(1)   |
O(n)   |    O(n)   |    O(1)   |    O(1)   |       O(n)       |
        +--------------------+-----------+-----------+-----------+-----------+----
-------+-----------+-----------+-----------+------------------+
        | Singly linked list |    0(n)   |    0(n)   |    0(1)   |    0(1)   |
O(n)   |    O(n)   |    O(1)   |    O(1)   |       O(n)       |
        +--------------------+-----------+-----------+-----------+-----------+----
-------+-----------+-----------+-----------+------------------+
        | Doubly Linked List |    0(n)   |    0(n)   |    0(1)   |    0(1)   |
O(n)   |    O(n)   |    O(1)   |    O(1)   |       O(n)       |
        +--------------------+-----------+-----------+-----------+-----------+----
-------+-----------+-----------+-----------+------------------+
        |      Skip List     | 0(log(n)) | 0(log(n)) | 0(log(n)) | 0(log(n)) |
O(n)   |    O(n)   |    O(n)   |    O(n)   |    O(n log(n))   |
        +--------------------+-----------+-----------+-----------+-----------+----
-------+-----------+-----------+-----------+------------------+
        |     Hash Table     |    N/A    |    O(1)   |    O(1)   |    O(1)   |
N/A    |    O(n)   |    O(n)   |    O(n)   |       O(n)       |
        +--------------------+-----------+-----------+-----------+-----------+----
-------+-----------+-----------+-----------+------------------+
        | Binary Search Tree | 0(log(n)) | 0(log(n)) | 0(log(n)) | 0(log(n)) |
O(n)   |    O(n)   |    O(n)   |    O(n)   |       O(n)       |
        +--------------------+-----------+-----------+-----------+-----------+----
-------+-----------+-----------+-----------+------------------+
        |   Cartesian Tree   |    N/A    | 0(log(n)) | 0(log(n)) | 0(log(n)) |
N/A    |    O(n)   |    O(n)   |    O(n)   |       O(n)       |
        +--------------------+-----------+-----------+-----------+-----------+----
-------+-----------+-----------+-----------+------------------+
        |       B-Tree       | 0(log(n)) | 0(log(n)) | 0(log(n)) | 0(log(n)) |
O(log(n)) | O(log(n)) | O(log(n)) | O(log(n)) |       O(n)       |
        +--------------------+-----------+-----------+-----------+-----------+----
-------+-----------+-----------+-----------+------------------+
        |   Read-Black Tree  | 0(log(n)) | 0(log(n)) | 0(log(n)) | 0(log(n)) |
O(log(n)) | O(log(n)) | O(log(n)) | O(log(n)) |       O(n)       |
        +--------------------+-----------+-----------+-----------+-----------+----
-------+-----------+-----------+-----------+------------------+
        |     Splay Tree     |    N/A    | 0(log(n)) | 0(log(n)) | 0(log(n)) |
N/A    | O(log(n)) | O(log(n)) | O(log(n)) |       O(n)       |
        +--------------------+-----------+-----------+-----------+-----------+----
-------+-----------+-----------+-----------+------------------+
        |      AVL Tree      | 0(log(n)) | 0(log(n)) | 0(log(n)) | 0(log(n)) |
O(log(n)) | O(log(n)) | O(log(n)) | O(log(n)) |       O(n)       |
        +--------------------+-----------+-----------+-----------+-----------+----
-------+-----------+-----------+-----------+------------------+
        |       KD Tree      | 0(log(n)) | 0(log(n)) | 0(log(n)) | 0(log(n)) |
```

```
O(n)   |    O(n)   |    O(n)   |    O(n)   |       O(n)       |
        +--------------------+----------+----------+----------+----------+----
-------+----------+----------+----------+------------------+
```

Sorting Algorithm

```
        +---------------+-------------------------------------------------+--------
----------+
        |    Algorithm   |               Time Complexity                | Space
Complexity |
        +---------------+-------------+----------------+--------------+--------
----------+
        |                |     Best    |     Average     |     Worst     |
Worst      |
        +---------------+-------------+----------------+--------------+--------
----------+
        | Quicksort      | Ω(n log(n)) | θ(n log(n))     | O(n^2)        | O(n
log(n))      |
        +---------------+-------------+----------------+--------------+--------
----------+
        | Mergesort      | Ω(n log(n)) | θ(n log(n))     | O(n log(n))   | O(n)
|
        +---------------+-------------+----------------+--------------+--------
----------+
        | Timesort       | Ω(n)        | θ(n log(n))     | O(n log(n))   | O(n)
|
        +---------------+-------------+----------------+--------------+--------
----------+
        | Heapsort       | Ω(n log(n)) | θ(n log(n))     | O(n log(n))   | O(1)
|
        +---------------+-------------+----------------+--------------+--------
----------+
        | Bubble sort    | Ω(n)        | θ(n ^ 2)        | O(n^2)        | O(1)
|
        +---------------+-------------+----------------+--------------+--------
----------+
        | Insertion sort | Ω(n)        | θ(n ^ 2)        | O(n^2)        | O(1)
|
        +---------------+-------------+----------------+--------------+--------
----------+
        | Selection Sort | Ω(n ^ 2)    | θ(n ^ 2)        | O(n^2)        | O(1)
|
        +---------------+-------------+----------------+--------------+--------
----------+
        | Tree sort      | Ω(n log(n)) | θ(n log(n))     | O(n^2)        | O(n)
|
        +---------------+-------------+----------------+--------------+--------
----------+
        | Shell sort     | Ω(n log(n)) | θ(n log(n) ^ 2) | O(n(log(n)^2) | O(1)
```

```
|
        +----------------+-------------+-----------------+---------------+--------
----------+
        | Bucket sort    | Ω (n + k )  | Θ(n + K)        | O(n^2)        | O(n)
|
        +----------------+-------------+-----------------+---------------+--------
----------+
        | Radix sort     | Ω (nk )     | Θ(n + K)        | O(nk)         | O(n+k)
|
        +----------------+-------------+-----------------+---------------+--------
----------+
        | Counting sort  | Ω (n + k )  | Θ(n + K)        | O(n+k)        | O(k)
|
        +----------------+-------------+-----------------+---------------+--------
----------+
        | Cube sort      | Ω (n)       | Θ(n log(n))     | O(n log(n))   | O(n)
|
        +----------------+-------------+-----------------+---------------+--------
----------+
        |                |             |                 |               |
|
        +----------------+-------------+-----------------+---------------+--------
----------+
```

Interview Questions

1. Difference between file structure and structure storage structure? he key difference between both the data structure is the memory area that is being accessed. When dealing with the structure that resides the main memory of the computer system, this is referred to as storage structure. When dealing with an auxiliary structure, we refer to it as file structures.

2. What is LIFO? LIFO is a short form of Last In First Out. It refers how data is accessed, stored and retrieved. Using this scheme, data that was stored last should be the one to be extracted first. This also means that in order to gain access to the first data, all the other data that was stored before this first data must first be retrieved and extracted.

3. What is FIFO? FIFO stands for First-in, First-out, and is used to represent how data is accessed in a queue. Data has been inserted into the queue list the longest is the one that is removed first.

4. What are common types of Data Structures?

    1. Array
    2. Queue
    3. Linked list
    4. Heap
    5. Tree
    6. Stack
    7. graph

5. What is a stack? A stack is a data structure in which only the top element can be accessed. As data is stored in the stack, each data is pushed downward, leaving the most recently added data on top.

6. What is a queue? A queue is a data structure that can simulate a list or stream of data. In this structure, new elements are inserted at one end, and existing elements are removed from the other end.

7. What is a binary tree? A binary search tree stores data in such a way that they can be retrieved very efficiently. The left subtree contains nodes whose keys are less than the node's key value, while the right subtree contains nodes whose keys are greater than or equal to the node's key value. Moreover, both subtrees are also binary search trees.

8. What data types are applied using recursion? Recursion, is a function that calls itself based on a terminating condition, makes use of the stack. Using LIFO, a call to a recursive function saves the return address so that it knows how to return to the calling function after the call terminates.

9. what are multidimensional arrays? Multidimensional arrays make use of multiple indexes to store data. It is useful when storing data that cannot be represented using single dimensional indexing, such as data representation in a board game, tables with data stored in more than one column.

10. linked list are linear or non linear in nature? Explain It depends on where you intend to apply linked lists. If you based it on storage, a linked list is considered non-linear. On the other hand, if you based it on access strategies, then a linked list is considered linear.

11. what is a heap? A Heap is a special Tree-based data structure in which the tree is a complete binary tree. Generally, Heaps can be of two types:

    Max-Heap: In a Max-Heap the key present at the root node must be greatest among the keys present at all of it's children. The same property must be recursively true for all sub-trees in that Binary Tree.

    Min-Heap: In a Min-Heap the key present at the root node must be minimum among the keys present at all of it's children. The same property must be recursively true for all sub-trees in that Binary Tree.

12. what is an orderd list? An ordered list is a list in which each node's position in the list is determined by the value of its key component, so that the key values form an increasing sequence, as the list is traversed.

13. What is a liner search? A linear search refers to the way a target key is being searched in a sequential data structure. In this method, each element in the list is checked and compared against the target key. The process is repeated until found or if the end of the file has been reached.

14. Difference between BFS and DFS? | Sr.No | BFS | DFS | |-------|----------|----------| | 1 | BFS stands for Breadth First Search. | DFS stands for Depth First Search. | | 2 | BFS(Breadth First Search) uses Queue data structure for finding the shortest path. | DFS(Depth First Search) uses Stack data structure. | | 3 | BFS can be used to find single source shortest path in an unweighted graph, because in BFS, we reach a vertex with minimum number of edges from a source vertex. | In DFS, we might traverse through more edges to reach a destination vertex from a source. | | 4 | BFS is more suitable for searching vertices which are closer to the given source. | DFS is more suitable when there are solutions away from source. | | 5 | BFS considers all

neighbors first and therefore not suitable for decision making trees used in games or puzzles. | DFS is more suitable for game or puzzle problems. We make a decision, then explore all paths through this decision. And if this decision leads to win situation, we stop. | | 6 | The Time complexity of BFS is O(V + E) when Adjacency List is used and O(V^2) when Adjacency Matrix is used, where V stands for vertices and E stands for edges. | The Time complexity of DFS is also O(V + E) when Adjacency List is used and O(V^2) when Adjacency Matrix is used, where V stands for vertices and E stands for edges. |

15. Heap vs stack? The heap is more flexible than the stack. That's because memory space for the heap can be dynamically allocated and de-allocated as needed. However, the memory of the heap can at times be slower when compared to that stack.

16. How does bubble sort, selection sort and merge sort work ?

17. How does signed and unsigned numbers affect memory? In the case of signed numbers, the first bit is used to indicate whether positive or negative, which leaves you with one bit short. With unsigned numbers, you have all bits available for that number. The effect is best seen in the number range (an unsigned 8-bit number has a range 0-255, while the 8-bit signed number has a range -128 to +127.

18. What is the realtionship between queue and linked list? The structre of queue is based on singly linked list which is dynamic in nature.

19. Give a basic algorithm for searching a binary search tree.

    1. if the tree is empty, then the target is not in the tree, end search
    2. if the tree is not empty, the target is in the tree
    3. check if the target is in the root item
    4. if a target is not in the root item, check if a target is smaller than the root's value
    5. if a target is smaller than the root's value, search the left subtree
    6. else, search the right subtree

20. what is a graph? A graph is one type of data structure that contains a set of ordered pairs. These ordered pairs are also referred to as edges or arcs and are used to connect nodes where data can be stored and retrieved.

21. linear vs non linear data structures? The linear data structure is a structure wherein data elements are adjacent to each other. Examples of linear data structure include arrays, linked lists, stacks, and queues. On the other hand, a non-linear data structure is a structure wherein each data element can connect to more than two adjacent data elements. Examples of nonlinear data structure include trees and graphs.

22. what is an avl tree? An AVL tree is a type of binary search tree that is always in a state of partially balanced. The balance is measured as a difference between the heights of the subtrees from the root. This self-balancing tree was known to be the first data structure to be designed as such.

23. what is a doubly linked list? Doubly linked lists are a special type of linked list wherein traversal across the data elements can be done in both directions. This is made possible by having two links in every node, one that links to the next node and another one that connects to the previous node

24. What is huffman algorith? Huffman's algorithm is used for creating extended binary trees that have minimum weighted path lengths from the given weights. It makes use of a table that contains the frequency of occurrence for each data element.

25. what is fibonacci search? Fibonacci search is a search algorithm that applies to a sorted array. It makes use of a divide-and-conquer approach that can significantly reduce the time needed in order to reach the target element.

26. Explain recursion? Recursive algorithm targets a problem by dividing it into smaller, manageable sub-problems. The output of one recursion after processing one sub-problem becomes the input to the next recursive process.

27. What is hashing? Hashing is a technique or process of mapping keys, values into the hash table by using a hash function. It is done for faster access to elements. The efficiency of mapping depends on the efficiency of the hash function used.

    Let a hash function H(x) maps the value x at the index x%10 in an Array. For example if the list of values is [11,12,13,14,15] it will be stored at positions {1,2,3,4,5} in the array or Hash table respectively.

28. What is a spanning tress? A spanning tree is a subset of Graph G, which has all the vertices covered with minimum possible number of edges. A spanning tree does not have cycles and it can not be disconnected.

29. What are greedy algorith,? The below given problems find their solution using greedy algorithm approach –

    ```
    Travelling Salesman Problem
    Prim's Minimal Spanning Tree Algorithm
    Kruskal's Minimal Spanning Tree Algorithm
    Dijkstra's Minimal Spanning Tree Algorithm
    Graph - Map Coloring
    Graph - Vertex Cover
    Knapsack Problem
    Job Scheduling Problem
    ```

30. what is an algorithm? Algorithm is a step by step procedure, which defines a set of instructions to be executed in certain order to get the desired output.

31. which DS should be used for implementiong LRU cache? We use two data structures to implement an LRU Cache.

    Queue which is implemented using a doubly linked list. The maximum size of the queue will be equal to the total number of frames available (cache size). The most recently used pages will be near rear end and least recently pages will be near front end.

    A Hash with page number as key and address of the corresponding queue node as value. See How to implement LRU caching scheme? What data structures should be used?

Database Interview Questions

Most common Interview Questions

1. What is RDBMS?
2. What are records?
3. What are advantages of RDBMS?
4. What is Data Redundancy?
5. What are Database Relationships?
    1. 1 - 1
    2. 1 - many
    3. many - 1
6. Explain Normalization and De-normalization?
7. Why is indexing used?
8. Types of SQL Statements?
    1. DDL: Data Definition Language
    2. DML: Data Manipulation language
    3. DCL: Data Control Language
9. State DDL , DML and DCL Clauses?
    1. DDL
        1. Create
        2. Alter
        3. Truncate
        4. Drop
        5. Rename
    2. DML
        1. Insert
        2. Update
        3. Delete
        4. Merge
    3. DCL
        1. Commit
        2. Rollback
        3. Savepoint
10. Difference between Having and Where Clause?
11. Explain Indexing and its purpose?
12. What are views?
13. What are cursors and its type?
14. What are database transactions?
15. What are database lock?
16. Define Joins and explain its types?
17. What are aggregated functions?
18. What are keys?
19. Difference between delete and truncate and drop?
20. What is normalization and its types?
21. Find Second Highest Salary?
    1. Select Max(salary) from employee order by salary DESC n-1, 1;