**Important Oral Question for Data Structure Lab Practical Exam**

1. **What is a Data Structure?**
   A data structure is a way of organizing the data so that the data can be used efficiently. Different kinds of data structures are suited to different kinds of applications, and some are highly specialized to specific tasks. For example, B-trees are particularly well-suited for the implementation of databases, while compiler implementations usually use hash tables to look up identifiers.
2. **What are linear and non-linear data Structures?**

- **Linear:** A data structure is said to be linear if its elements form a sequence or a linear list. Examples: Array. Linked List, Stacks and Queues
- **Non-Linear:** A data structure is said to be non-linear if the traversal of nodes is nonlinear in nature. Example: Graph and Trees.

3. **What are the various operations that can be performed on different Data Structures?**

- **Insertion** ? Add a new data item in the given collection of data items.
- **Deletion** ? Delete an existing data item from the given collection of data items.
- **Traversal** ? Access each data item exactly once so that it can be processed.
- **Searching** ? Find out the location of the data item if it exists in the given collection of data items.
- **Sorting** ? Arranging the data items in some order i.e. in ascending or descending order in case of numerical data and in dictionary order in case of alphanumeric data.

4. **How is an Array different from Linked List?**

- The size of the arrays is fixed, Linked Lists are Dynamic in size.
- Inserting and deleting a new element in an array of elements is expensive, Whereas both insertion and deletion can easily be done in Linked Lists.
- Random access is not allowed in Linked Listed.
- Extra memory space for a pointer is required with each element of the Linked list.
- Arrays have better cache locality that can make a pretty big difference in performance.

5. **What is an array?**

- Arrays are the collection of **similar** types of data stored at **contiguous** memory locations.
- It is the simplest data structure where the data element can be accessed randomly just by using its index number.

**6. What is a multidimensional array?**

- Multi-dimensional arrays are those data structures that span across more than one dimension.
- This indicates that there will be more than one index variable for every point of storage. This type of data structure is primarily used in cases where data cannot be represented or stored using only one dimension. Most commonly used multidimensional arrays are 2D arrays.

**2D arrays emulates the tabular form structure which provides ease of holding the bulk of data that are accessed using row and column pointers.**

**7.What is a linked list?**

A linked list is a data structure that has **sequence of nodes** where every node is connected to the next node by means of a reference pointer. The elements are **not stored in adjacent** memory locations. They are linked using pointers to form a chain. This forms a chain-like link for data storage.

- Each node element has two parts:
- a data field
- a reference (or pointer) to the next node.

- The first node in a linked list is called the head and the last node in the list has the pointer to NULL. Null in the reference field indicates that the node is the last node. When the list is empty, the head is a null reference.
-

**8. What is Stack and where it can be used?**

Stack is a linear data structure which the order LIFO(Last In First Out) or FILO(First In Last Out) for accessing elements. Basic operations of the stack are: **Push, Pop, Peek**

Applications of Stack:

1. Infix to Postfix Conversion using Stack
2. Evaluation of Postfix Expression
3. Reverse a String using Stack
4. Implement two stacks in an array
5. Check for balanced parentheses in an expression

**What is a Queue, how it is different from the stack and how is it implemented?**

Queue is a linear structure that follows the order is **F**irst **I**n **F**irst **O**ut (FIFO) to access elements. Mainly the following are basic operations on queue: **Enqueue, Dequeue**, **Front, Rear**

The difference between stacks and queues is in removing. In a stack we remove the item the most recently added; in a queue, we remove the item the least recently added. Both Queues and Stacks can be implemented using Arrays and Linked Lists.

**9 What is a Linked List and What are its types?**

A linked list is a linear data structure (like arrays) where each element is a separate object. Each element (that is node) of a list is comprising of two items – the data and a reference to the next node.Types of Linked List :

1. **Singly Linked List :** In this type of linked list, every node stores address or reference of next node in list and the last node has next address or reference as NULL. For example 1->2->3->4->NULL
2. **Doubly Linked List :** Here, here are two references associated with each node, One of the reference points to the next node and one to the previous node. Eg. NULL<-1<->2<->3->NULL
3. **Circular Linked List :** Circular linked list is a linked list where all nodes are connected to form a circle. There is no NULL at the end. A circular linked list can be a singly circular linked list or doubly circular linked list. Eg. 1->2->3->1 [The next pointer of last node is pointing to the first]

10. **What is a data structure?**

The data structure is the way data is organized (stored) and manipulated for retrieval and access. It also defines the way different sets of data relate to one another, establishing relationships and forming algorithms.

11 **What is a linear data structure? Name a few examples.**

A data structure is linear if all its elements or data items are arranged in a sequence or a linear order. The elements are stored in a non-hierarchical way so that each item has successors and predecessors except the first and last element in the list.

**12 Can you tell how linear data structures differ from non-linear data structures?**

- If the elements of a data structure result in a sequence or a linear list then it is called a linear data structure. Whereas, traversal of nodes happens in a non-linear fashion in non-linear data structures.

- Lists, stacks, and queues are examples of linear data structures whereas graphs and trees are the examples of non-linear data structures.

13**. What are some applications of data structures**?

Numerical analysis, operating system, AI, compiler design, database management, graphics, statistical analysis, and simulation.

14**. What is the difference between file structure and storage structure?**

The difference lies in the memory area accessed. Storage structure refers to the data structure in the memory of the computer system, whereas file structure represents the storage structure in the auxiliary memory.

15. **What is a linked list data structure?**

It's a linear data structure or a sequence of data objects where elements are not stored in adjacent memory locations. The elements are linked using pointers to form a chain. Each element is a separate object, called a node. Each node has two items: a data field and a reference to the next node. The entry point in a linked list is called the head. Where the list is empty, the head is a null reference and the last node has a reference to null.

A linked list is a dynamic data structure, where the number of nodes is not fixed, and the list has the ability to grow and shrink on demand.

It is applied in cases where:

- We deal with an unknown number of objects or don't know how many items are in the list
- We need constant-time insertions/deletions from the list, as in real-time computing where time predictability is critical
- Random access to any elements is not needed
- The algorithm requires a data structure where objects need to be stored irrespective of their physical address in memory
- We need to insert items in the middle of the list as in a priority queue

Some implementations are stacks and queues, graphs, directory of names, dynamic memory allocation, and performing arithmetic operations on long integers.

16. **Are linked lists considered linear or non-linear data structures?**

Linked lists are considered both linear and non-linear data structures depending upon the application they are used for. When used for access strategies, it is considered as a linear data-structure. When used for data storage, it is considered a non-linear data structure.

17. **What are the advantages of a linked list over an array? In which scenarios do we use Linked List and when Array?**

Advantages of a linked list over an array are:

1. Insertion and Deletion

Insertion and deletion of nodes is an easier process, as we only update the address present in the next pointer of a node. It's expensive to do the same in an array as the room has to be created for the new elements and existing elements must be shifted.

2. Dynamic Data Structure

As a linked list is a dynamic data structure, there is no need to give an initial size as it can grow and shrink at runtime by allocating and deallocating memory. However, the size is limited in an array as the number of elements is statically stored in the main memory.

3. No wastage of memory

As the size of a linked list can increase or decrease depending on the demands of the program, and memory is allocated only when required, there is no memory wasted. In the case of an array, there is memory wastage. For instance, if we declare an array of size 10 and store only five elements in it, then the space for five elements is wasted.

4. Implementation

Data structures like stack and queues are more easily implemented using a linked list than an array.

Some scenarios where we use linked list over array are:

- When we know the upper limit on the number of elements in advance
- When there are a large number of add or remove operations
- When there are no large number of random access to elements
- When we want to insert items in the middle of the list, such as when implementing a priority queue

Some scenarios in which we use array over the linked list are:

- When we need to index or randomly access elements
- When we know the number of elements in the array beforehand, so we can allocate the correct amount of memory

- When we need speed when iterating through all the elements in the sequence
- When memory is a concern; filled arrays use less memory than linked lists, as each element in the array is the data but each linked list node requires the data as well as one or more pointers to the other elements in the linked list

In summary, we consider the requirements of space, time, and ease of implementation to decide whether to use a linked list or array.

18. **What is a doubly-linked list? Give some examples**.

It is a complex type (double-ended LL) of a linked list in which a node has two links, one that connects to the next node in the sequence and another that connects to the previous node. This allows traversal across the data elements in both directions.

Examples include:

- A music playlist with next and previous navigation buttons
- The browser cache with BACK-FORWARD visited pages
- The undo and redo functionality on a browser, where you can reverse the node to get to the previous page

19. **How do you reference all of the elements in a one-dimension array?**

All of the elements in a one-dimension array can be referenced using an indexed loop as the array subscript so that the counter runs from 0 to the array size minus one.

20. **What are dynamic data structures? Name a few.**

They are collections of data in memory that expand and contract to grow or shrink in size as a program runs. This enables the programmer to control exactly how much memory is to be utilized.

Examples are the dynamic array, linked list, stack, queue, and heap.

.

21. **What is an algorithm?**

An algorithm is a step by step method of solving a problem or manipulating data. It defines a set of instructions to be executed in a certain order to get the desired output.

22**. Why do we need to do an algorithm analysis**?

A problem can be solved in more than one way using several solution algorithms. Algorithm analysis provides an estimation of the required resources of an algorithm to solve a specific computational problem. The amount of time and space resources required to execute is also determined.

The time complexity of an algorithm quantifies the amount of time taken for an algorithm to run as a function of the length of the input. The space complexity quantifies the amount of space or memory taken by an algorithm, to run as a function of the length of the input.

23. **What is a stack?**

A stack is an abstract data type that specifies a linear data structure, as in a real physical stack or piles where you can only take the top item off the stack in order to remove things. Thus, insertion (push) and deletion (pop) of items take place only at one end called top of the stack, with a particular order: LIFO (Last In First Out) or FILO (First In Last Out).

24. Where are stacks used?

- Expression, evaluation, or conversion of evaluating prefix, postfix, and infix expressions
- Syntax parsing
- String reversal
- Parenthesis checking
- Backtracking

25. **What is a queue data structure?**

A queue is an abstract data type that specifies a linear data structure or an ordered list, using the First In First Out (FIFO) operation to access elements. Insert operations can be performed only at one end called REAR and delete operations can be performed only at the other end called FRONT.

26**. List some applications of queue data structure.**

To prioritize jobs as in the following scenarios:

- As waiting lists for a single shared resource in a printer, CPU, call center systems, or image uploads; where the first one entered is the first to be processed
- In the asynchronous transfer of data; or example pipes, file IO, and sockets
- As buffers in applications like MP3 media players and CD players
- To maintain the playlist in media players (to add or remove the songs)

27. **What is a Dequeue?**

It is a double-ended queue, or a data structure, where the elements can be inserted or deleted at both ends (FRONT and REAR).

28. **What operations can be performed on queues?**

- enqueue() adds an element to the end of the queue
- dequeue() removes an element from the front of the queue
- init() is used for initializing the queue
- isEmpty tests for whether or not the queue is empty
- The front is used to get the value of the first data item but does not remove it
- The rear is used to get the last item from a queue

**29. What is a priority queue?**

- A priority queue is an abstract data type that is like a normal queue but has priority assigned to elements.
- Elements with higher priority are processed before the elements with a lower priority.
- In order to implement this, a minimum of two queues are required - one for the data and the other to store the priority.

30**. Where can stack data structure be used?**

- Expression evaluation
- Backtracking
- Memory management
- Function calling and return

31**. What is the difference between a PUSH and a POP?**

The acronyms stand for Pushing and Popping operations performed on a stack. These are ways data is stored and retrieved.

- PUSH is used to add an item to a stack, while POP is used to remove an item.
- PUSH takes two arguments, the name of the stack to add the data to and the value of the entry to be added. POP only needs the name of the stack.

- When the stack is filled and another PUSH command is issued, you get a stack overflow error, which means that the stack can no longer accommodate the last PUSH. In POP, a stack underflow error occurs when you're trying to POP an already empty stack.

32. **Which sorting algorithm is considered the fastest? Why?**

A single sorting algorithm can't be considered best, as each algorithm is designed for a particular data structure and data set. However, the QuickSort algorithm is generally considered the fastest because it has the best performance for most inputs.

Its advantages over other sorting algorithms include the following:

- Cache-efficient: It linearly scans and linearly partitions the input. This means we can make the most of every cache load.
- Can skip some swaps: As QuickSort is slightly sensitive to input that is in the right order, it can skip some swaps.
- Efficient even in worst-case input sets, as the order is generally random.
- Easy adaption to already- or mostly-sorted inputs.
- When speed takes priority over stability.

33. **What is the merge sort? How does it work?**

Merge sort is a divide-and-conquer algorithm for sorting the data. It works by merging and sorting adjacent data to create bigger sorted lists, which are then merged recursively to form even bigger sorted lists until you have one single sorted list.

34**. How does the Selection sort work?**

Selection sort works by repeatedly picking the smallest number in ascending order from the list and placing it at the beginning. This process is repeated moving toward the end of the list or sorted subarray.

Scan all items and find the smallest. Switch over the position as the first item. Repeat the selection sort on the remaining N-1 items. We always iterate forward (i from 0 to N-1) and swap with the smallest element (always i).

Time complexity: best case O(n2); worst O(n2)

Space complexity: worst O(1)

**35. What are the advantages of binary search over a linear search?**

In a sorted list:

- A binary search is more efficient than a linear search because we perform fewer comparisons. With linear search, we can only eliminate one element per comparison each time we fail to find the value we are looking for, but with the binary search, we eliminate half the set with each comparison.
- Binary search runs in O(log n) time compared to linear search's O(n) time. This means that the more of the elements present in the search array, the faster is binary search compared to a linear search.

**36. Do dynamic memory allocations help in managing data? How?**

Dynamic memory allocation stores simple structured data types at runtime. It has the ability to combine separately allocated structured blocks to form composite structures that expand and contract as needed, thus helping manage data of data blocks of arbitrary size, in arbitrary order.

**37. Which data structure is used to perform recursion?**

Stack data structure is used in recursion due to its last in first out nature. Operating system maintains the stack in order to save the iteration variables at each function call

**38. Write the stack overflow condition.**

Overflow occurs when **top = Maxsize -1**

**39. What is the difference between PUSH and POP?**

PUSH and POP operations specify how data is stored and retrieved in a stack.

**PUSH:** PUSH specifies that data is being "inserted" into the stack.

**POP:** POP specifies data retrieval. It means that data is being deleted from the stack.

---

**40. Write the steps involved in the insertion and deletion of an element in the stack**.

**Push:**

- Increment the variable top so that it can refer to the next memory allocation
- Copy the item to the at the array index value equal to the top
- Repeat step 1 and 2 until stack overflows

**Pop:**

- Store the topmost element into the an another variable
- Decrement the value of the top

- Return the topmost element

41What is a postfix expression?

An expression in which operators follow the operands is known as postfix expression. The main benefit of this form is that there is no need to group sub-expressions in parentheses or to consider operator precedence.

The expression "a + b" will be represented as "ab+" in postfix notation.

42. How are the elements of a 2D array are stored in the memory?

There are two techniques by using which, the elements of a 2D array can be stored in the memory.

- **Row-Major Order:** In row-major ordering, all the rows of the 2D array are stored into the memory contiguously. First, the 1st row of the array is stored into the memory completely, then the 2nd row of the array is stored into the memory completely and so on till the last row.
- **Column-Major Order:** In column-major ordering, all the columns of the 2D array are stored into the memory contiguously. first, the 1st column of the array is stored into the memory completely, then the 2nd row of the array is stored into the memory completely and so on till the last column of the array.

---

**43 Calculate the address of a random element present in a 2D array, given base address as BA**.

**Row-Major Order:** If array is declared as a[m][n] where m is the number of rows while n is the number of columns, then address of an element a[i][j] of the array stored in row major order is calculated as,

**Address(a[i][j]) = B. A. + (i * n + j) * size**

**Column-Major Order:** If array is declared as a[m][n] where m is the number of rows while n is the number of columns, then address of an element a[i][j] of the array stored in column major order is calculated as

**Address(a[i][j]) = ((j*m)+i)*Size + BA**.

**44.List some applications of queue data structure.**

The Applications of the queue is given as follows:

- Queues are widely used as waiting lists for a single shared resource like a printer, disk, CPU.
- Queues are used in the asynchronous transfer of data (where data is not being transferred at the same rate between two processes) for eg. pipes, file IO, sockets.
- Queues are used as buffers in most of the applications like MP3 media player, CD player, etc.
- Queues are used to maintain the playlist in media players to add and remove the songs from the play-list.
- Queues are used in operating systems for handling interrupts.

---

**45. What are the drawbacks of array implementation of Queue?**

- **Memory Wastage:** The space of the array, which is used to store queue elements, can never be reused to store the elements of that queue because the elements can only be inserted at front end and the value of front might be so high so that, all the space before that, can never be filled.
- **Array Size:** There might be situations in which, we may need to extend the queue to insert more elements if we use an array to implement queue, It will almost be impossible to extend the array size, therefore deciding the correct array size is always a problem in array implementation of queue.

---

**46. What are the scenarios in which an element can be inserted into the circular queue?**

- If (rear + 1)%maxsize = front, the queue is full. In that case, overflow occurs and therefore, insertion can not be performed in the queue.
- If rear != max - 1, the rear will be incremented to the mod(maxsize) and the new value will be inserted at the rear end of the queue.
- If front != 0 and rear = max - 1, it means that queue is not full therefore, set the value of rear to 0 and insert the new element there.

---

**47.What is a dequeue?**

Dequeue (also known as double-ended queue) can be defined as an ordered set of elements in which the insertion and deletion can be performed at both the ends, i.e. front and rear.

---

**48. What is the minimum number of queues that can be used to implement a priority queue?**

Two queues are needed. One queue is used to store the data elements, and another is used for storing priorities.

**49.** What is the difference between NULL and VOID?

- Null is actually a value, whereas Void is a data type identifier.
- A null variable simply indicates an empty value, whereas void is used to identify pointers as having no initial size.

**50.When is a binary search best applied?**

- A binary search is an algorithm that is best applied to search a list when the elements are already in order or sorted. The list is searched starting in the middle, such that if that middle value is not the target search key, it will check to see if it will continue the search on the lower half of the list or the higher half. The split and search will then continue in the same manner.

**51.** Wha**t is merge sort?**

Merge sort, is a  divide-and-conquer approach for sorting the data. In a sequence of data, adjacent ones are merged and sorted to create bigger sorted lists. These sorted lists are then merged again to form an even bigger sorted list, which continues until you have one single sorted list.

**52.  What is a linear 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.

**53.  How does variable declaration affect memory allocation?**

The amount of memory to be allocated or reserved would depend on the data type of the variable being declared. For example, if a variable is declared to be of integer type, then 32 bits of memory storage will be reserved for that variable

**54**.  **Differentiate STACK from ARRAY.**

Stack follows a LIFO pattern. It means that data access follows a sequence wherein the last data to be stored when the first one to be extracted. Arrays, on the other hand, does not follow a particular order and instead can be accessed by referring to the indexed element within the array.

**55. What is a bubble sort and how do you perform it?**

A bubble sort is one sorting technique that can be applied to data structures such as an array. It works by comparing adjacent elements and exchanges their values if they are out of order. This method lets the smaller values "bubble" to the top of the list, while the larger value sinks to the bottom.

**56. What are the parts of a linked list?**

A linked list typically has two parts: the head and the tail. Between the head and tail lie the actual nodes. All these nodes are linked sequentially.

**57. How does selection sort work?**

Selection sort works by picking the smallest number from the list and placing it at the front. This process is repeated for the second position towards the end of the list. It is the simplest sort algorithm.

**58. What are doubly linked lists?**

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.

**59. 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.

**60. Briefly explain recursive algorithm.**

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

**61. How do you search for a target key in a linked list?**

To find the target key in a linked list, you have to apply sequential search. Each node is traversed and compared with the target key, and if it is different, then it follows the link to the next node. This traversal continues until either the target key is found or if the last node is reached.