Q1. what is linked List explain in sort.

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations.

Linked list is a linear data structure which is used to maintain a list-like structure in the computer memory. It is a gro
up
of nodes that are not stored at contiguous locations. Each node of the list is linked to its adjacent node with the help of pointers.

A node contains two fields i.e. data stored at that particular address and the pointer which contains the address of the next node in the memory.
The last node of the list contains pointer to the null.

In its simplest form, a linked list can be thought of as a kind of chain:
The "links" (or elements) in the linked-list "chain" are called nodes.

- Where Are They Used?
Linked lists can be used anywhere other linear data structures are used. In fact, they can be used to implement severa
l other common data types,
such as:
Stacks
Queues
Lists
Generally speaking, for any application that deals with an unknown number of data elements a linked list will need t
o be used.

Q2. application of linked list and doubly and singular and circular linked list. \

- Applications of linked list in computer science –

- Implementation of stacks and queues
- Implementation of graphs : Adjacency list representation of graphs is most popular which is uses linked list to st
ore adjacent vertices.
- Dynamic memory allocation : We use linked list of free blocks.
- Maintaining directory of names
- Performing arithmetic operations on long integers
- Manipulation of polynomials by storing constants in the node of linked list
- representing sparse matrices

- Applications of linked list in real world-

- Image viewer – Previous and next images are linked, hence can be accessed by next and previous button.
- Previous and next page in web browser – We can access previous and next url searched in web browser by pressi
ng back and next button since,
they are linked as linked list.
- Music Player – Songs in music player are linked to previous and next song. you can play songs either from starti
ng or ending of the list.

- Applications of Circular Linked Lists:

Useful for implementation of queue. Unlike this implementation, we don't need to maintain two pointers for front an
d rear if we use circular linked

list. We can maintain a pointer to the last inserted node and front can always be obtained as next of last.
Circular lists are useful in applications to repeatedly go around the list. For example, when multiple applications are running on a PC, it is
common for the operating system to put the running applications on a list and then to cycle through them, giving eac h of them a slice of time to
execute, and then making them wait while the CPU is given to another application. It is convenient for the operating system to use a circular list
so that when it reaches the end of the list it can cycle around to the front of the list.

- Circular Doubly Linked Lists are used for implementation of advanced data structures like Fibonacci Heap.

- An example problem:

Design a data structure that supports following operations efficiently.

- getMin : Gets minimum
- extractMin : Removes minimum
- getMax : Gets maximum
- extractMax : Removes maximum

- insert : Inserts an item. It may be assumed that the inserted item is always greater than maximum so far. For examp le, a valid insertion
order is 10, 12, 13, 20, 50.

Doubly linked list is the best solution here. We maintain head and tail pointers, since inserted item is always greatest , we insert at tail.
Deleting an item from head or tail can be done in O(1) time. So all operations take O(1) time.

Q3. Types of Linked Lists.

The following are the types of linked list:

Singly Linked list
Doubly Linked list
Circular Linked list
Doubly Circular Linked list
Header Linked List

Doubly-linked Lists
A doubly-linked list has two pointers and a single value. The second pointer is used as a reference backwards to the previous node in the list.
This means that the head and tail of the list both point to “empty” nodes. This solves some of the traversal challenges associated with
singly-linked lists, but also adds significantly to the memory required.

Circularly-Linked Lists
In a circularly-linked list the tail pointer references the head of the list. This means that without having to use reverse traversal we
can access any node in the list from any other node. without requiring any additional space in memory.

1. Singly Linked List:
It is the simplest type of linked list in which every node contains some data and a pointer to the next

node of the same
data type. The node contains a pointer to the next node means that the node stores the address of the next node in the sequence. A single linked
list allows traversal of data only in one way.

note:-

- Each node has a single link to another node is called Singly Linked List.
- Singly Linked List does not store any pointer any reference to the previous node.
- Each node stores the contents of the node and a reference to the next node in the list.
- In a singly linked list, last node has a pointer which indicates that it is the last node. It requires a reference to the first
  node to store a single linked list.
- It has two successive nodes linked together in linear way and contains address of the next node to be followed.
- It has successor and predecessor. First node does not have predecessor while last node does not have successor. Last node have
  successor reference as NULL.
- It has only single link for the next node.
- In this type of linked list, only forward sequential movement is possible, no direct access is allowed.

2. Doubly Linked List:
A doubly linked list or a two-way linked list is a more complex type of linked list which contains a pointer to the next
as well as the previous node in sequence, Therefore, it contains three parts are data, a pointer to the next node, and a pointer to the previous
node. This would enable us to traverse the list in the backward direction as well.

note:-
- Doubly linked list is a sequence of elements in which every node has link to its previous node and next node.
- Traversing can be done in both directions and displays the contents in the whole list.

- Important Note:
First node is always pointed by head. In doubly linked list, previous field of the first node is always NULL (it must be NULL) and the next
field of the last must be NULL.

In the above figure we see that, doubly linked list contains three fields. In this, link of two nodes allow traversal of the list in either direction.
There is no need to traverse the list to find the previous node. We can traverse from head to tail as well as tail to head.

- Advantages of Doubly Linked List
    - Doubly linked list can be traversed in both forward and backward directions.
    - To delete a node in singly linked list, the previous node is required, while in doubly linked list, we can get the previous node using previous pointer.
    - It is very convenient than singly linked list. Doubly linked list maintains the links for bidirectional traversing.

- Disadvantages of Doubly Linked List
    - In doubly linked list, each node requires extra space for previous pointer.
    - All operations such as Insert, Delete, Traverse etc. require extra previous pointer to be maintained.

3. Circular Linked List:
A circular linked list is that in which the last node contains the pointer to the first node of the list. While traversing

a circular liked list, we can begin at any node and traverse the list in any direction forward and backward until we re
ach the same node we started.
Thus, a circular linked list has no beginning and no end.

note:-
- Circular linked list is similar to singly linked list. The only difference is that in circular linked list, the last nod
e points to the
first node in the list.
- It is a sequence of elements in which every element has link to its next element in the sequence and has a link
to the first element
in the sequence.

In the above figure we see that, each node points to its next node in the sequence but the last node points to the first
node in the list.
The previous element stores the address of the next element and the last element stores the address of the starting ele
ment. It forms a circular
chain because the element points to each other in a circular way.
- In circular linked list, the memory can be allocated when it is required because it has a dynamic size.
- Circular linked list is used in personal computers, where multiple applications are running. The operating syst
em provides a fixed
time slot for all running applications and the running applications are kept in a circular linked list until all the
applications
are completed. This is a real life example of circular linked list.
- We can insert elements anywhere in circular linked list, but in the array we cannot insert elements anywhere i
n the list because it
is in the contiguous memory.

4. Doubly Circular linked list:
A Doubly Circular linked list or a circular two-way linked list is a more complex type of linked-l
ist that
contains a pointer to the next as well as the previous node in the sequence. The difference between the doubly linked
and circular doubly
list is the same as that between a singly linked list and a circular linked list. The circular doubly linked list does not c
ontain null in
the previous field of the first node.

note:-

- Doubly circular linked list is a linked data structure which consists of a set of sequentially linked records calle
d nodes.
- Doubly circular linked list can be conceptualized as two singly linked lists formed from the same data items, b
ut in opposite
sequential orders.

The above diagram represents the basic structure of Doubly Circular Linked List. In doubly circular linked list, the p
revious link of the
first node points to the last node and the next link of the last node points to the first node.
- In doubly circular linked list, each node contains two fields called links used to represent references to the pre
vious and the
next node in the sequence of nodes.
5. Header Linked List:
A header linked list is a special type of linked list which contains a header node at the beginning of th
e list. So,
in a header linked list START will not point to the first node of the list but START will contain the address of the he
ader node.

Q4. Explain multiply linked list in short.

Solution:
Traverse both lists and generate the required numbers to be multiplied and then return the multiplied values of the tw o numbers.

- Algorithm to generate the number from linked list representation:

```
1) Initialize a variable to zero
2) Start traversing the linked list
3) Add the value of first node to this variable
4) From the second node, multiply the variable by 10
   and also take modulus of this value by 10^9+7
   and then add the value of the node to this
   variable.
5) Repeat step 4 until we reach the last node of the list.
```

Q4. how many pointers are necessary to implement a simple linked list.

For a singly-linked list, you need one pointer to keep track of where it starts, and one pointer in each link on the list.

For a doubly-linked list, you need two pointers to keep track of where it starts and where it ends, and two pointers in each link on the list.

note:-

N+1 pointers!

Where, N is the size of your linked list.

N pointers in each of the nodes and one head pointer that will allow you to perform any operation.

So, initially you need to make only one pointer(head) when your linked list is empty. As your linked list grows every node will contain a
pointer in order to add more nodes later on.

note 2:-

a linked list consists of a node which has two parts one is called the head, which stores the address to where the node points and the other
part which stores the data. so to implement a simple linked list you might require only one pointer.

But while performing an insertion function in the linked list then you might have to declare another temporary pointe r which stores the address
of the new node to be inserted.

note 3: -

If the list is empty, one pointer is enough:

Node head = null;

In general, 1 pointer (usually called next) per node in the list and 1 to point to the beginning of the list (usually referr ed to as head). Each node's pointer points to the next node in the list, last one points to null (but is still there).

note 4:-

For a single linked list:

1 pointer per element (next), therefore a total of N pointers.
However, it is often implemented that there is a pointer to the the head of the list which serves as the place holder for the list, therefore N+1 pointers

For a doubly linked list where you can traverse both forwards and backwards.

For each element you have a pointer for next and a pointer for prev.
That gives you a total of 2N pointers. If you have the place holder to the head of the list, you are looking at 2N+1 poi nters

Q5. how can you represent a linked list node?

Representation:
A linked list is represented by a pointer to the first node of the linked list. The first node is called the head. If the link ed list is empty,
then the value of the head is NULL.
Each node in a list consists of at least two parts:
1) data
2) Pointer (Or Reference) to the next node
In C, we can represent a node using structures. Below is an example of a linked list node with integer data.
In Java or C #, LinkedList can be represented as a class and a Node as a separate class. The LinkedList class contain s a reference of Node class type.

Q5. which type of memory allocation is referred to for liinked list.
Dynamic memory allocation is referred for Linked lists.
By dynamically allocating each node, you're only limited by your available memory.

The basic purpose of a linked list is to link nodes one after the other. Suppose you don't want to use dynamic memor y allocations, then I
suggest, don't go for linked list, instead you can use array of structures. Each struct will have two values, one for the data and another
for pointing to the next array index.

Dynamic allocations will help only if you don't know the number of inputs. Also it will help in memory management , in case you don't have
enough available contiguous memories. Deletion and insertions will also become easier than if you use array.

Q 66. what do you know about traversal in linked list?
Traversing is the most common operation that is performed in almost every scenario of singly linked list. Traversing means visiting each node of
the list once in order to perform some operation on that.

A data structure contains elements, which contain data
Traversing a data structure means: "visiting" or "touching" the elements of the structure, and doing something with the data
(Traversing is also sometimes called iterating over the data structure)
For example you could have a singly-linked list, with elements that are instances of this class:

```
 class LNode {
  Object data;
  LNode next;
 }
```

Suppose first is a pointer that points to the first element of a list of LNode objects; the last element in the list has a null next field

Traversing that list, from first to last, and printing out data in the elements, would print:
1 2 3

Q67. what are the main differences between the linked list and linear array.?

S. No.	Array	Linked List
1.	Insertions and deletions are difficult.	Insertions and deletions can be done easily.
2.	It needs movements of elements for insertion and deletion.	It does not need movement of nodes for insertion and deletion.
3.	In it space is wasted.	In it space is not wasted.
4.	It is more expensive.	It is less expensive.
5.	It requires less space as only information is stored.	It requires more space as pointers are also stored along with information.
6.	Its size is fixed.	Its size is not fixed.
7.	It can not be extended or reduced according to requirements.	It can be extended or reduced according to requirements.
8.	Same amount of time is required to access each element.	Different amount of time is required to access each element.
9.	Elements are stored in consecutive memory locations.	Elements may or may not be stored in consecutive memory locations.
10.	If have to go to a particular element then we can reach there directly.	If we have to go to a particular node then we have to go through all those nodes that come before that node.

Q67. mention few application of linked list?

note 1:-

- Linked Lists can be used to implement Stacks , Queues.
- Linked Lists can also be used to implement Graphs. (Adjacency list representation of Graph).
- Implementing Hash Tables :- Each Bucket of the hash table can itself be a linked list. (Open chain hashing).
- Undo functionality in Photoshop or Word . Linked list of states.
- A polynomial can be represented in an array or in a linked list by simply storing the coefficient and exponent of each term.

- However, for any polynomial operation , such as addition or multiplication of polynomials , linked list represent ation is more easier
  to deal with.
- Linked lists are useful for dynamic memory allocation.
- The real life application where the circular linked list is used is our Personal Computers, where multiple applicat ions are running.
- All the running applications are kept in a circular linked list and the OS gives a fixed time slot to all for running. The Operating
  System keeps on iterating over the linked list until all the applications are completed.

note 2:-
- It is used to implement stacks and queues which are like fundamental needs throughout computer science.
- To prevent the collision between the data in the hash map, we use a singly linked list.
- If we ever noticed the functioning of a casual notepad, it also uses a singly linked list to perform undo or redo or deleting functions.
- We can think of its use in a photo viewer for having look at photos continuously in a slide show.
- In the system of train, the idea is like a singly linked list, as if you want to add a Boggie, either you have to take a new boggie to
  add at last or you must spot a place in between boggies and add it.

Q68. what are the advantage of the linked list?

Advantages Of Linked List:

- Dynamic data structure: A linked list is a dynamic arrangement so it can grow and shrink at runtime by allocatin g and deallocating memory.
  So there is no need to give the initial size of the linked list.
- No memory wastage: In the Linked list, efficient memory utilization can be achieved since the size of the linked list increase or
  decrease at run time so there is no memory wastage and there is no need to pre-allocate the memory.
- Implementation: Linear data structures like stack and queues are often easily implemented using a linked list.
- Insertion and Deletion Operations: Insertion and deletion operations are quite easier in the linked list. There is no need to shift
  elements after the insertion or deletion of an element only the address present in the next pointer needs to be up dated.

Disadvantages Of Linked List:

- Memory usage: More memory is required in the linked list as compared to an array. Because in a linked list, a po inter is also required to \
  store the address of the next element and it requires extra memory for itself.
- Traversal: In a Linked list traversal is more time-consuming as compared to an array. Direct access to an element is not possible in a
  linked list as in an array by index. For example, for accessing a node at position n, one has to traverse all the no des before it.
- Reverse Traversing: In a singly linked list reverse traversing is not possible, but in the case of a doubly-linked lis t, it can be possible
  as it contains a pointer to the previously connected nodes with each node. For performing this extra memory is required for the back pointer hence, there is a wastage of memory.
- Random Access: Random access is not possible in a linked list due to its dynamic memory allocation.

Advantages of Linked List

- The linked list is a dynamic data structure.
- You can also decrease and increase the linked list at run-time. That is, you can allocate and deallocate memory at run-time itself.
- In this, you can easily do insertion and deletion functions. That is, you can easily insert and delete the node.
- Memory is well utilized in the linked list. Because in it, we do not have to allocate memory in advance.
- Its access time is very fast, and it can be accessed at a certain time without memory overhead.
- You can easily implement linear data structures using the linked list like a stack, queue.
-

Disadvantages of Linked List

-
- The linked list requires more memory to store the elements than an array, because each node of the linked list poi nts a pointer, due to
which it requires more memory.
- It is very difficult to traverse the nodes in a linked list. In this, we cannot access randomly to any one node. (As w e do in the array by
index.) For example: – If we want to traverse a node in an n position, then we have to traverse all the nodes that c ome before n, which will
spoil a lot of our time.
- Reverse traversing in a linked list is very difficult, because it requires more memory for the pointer.

- Application of Linked List

The linked list is a primitive data structure, which is used in various types of applications.

- It is used to maintain directory names.
- The linked list can perform arithmetic operations in the long integer.
- Polynomials can be manipulated by storing constant in the node of the linked list.
- We can also use it to next and previous images in the image viewer.
- With the help of the linked list, we can move songs back and forth in the music player.
- The linked list is also used for undo in word and Photoshop applications.
- All the running applications in the computer are stored in the circular linked list, and the operating system provid es them with a
fixed time slot.
- It can also be used to implement hash tables.

Advantages and Disadvantages of Linked List

- Advantages of Linked List

- Dynamic Data Structure

Linked list is a dynamic data structure so it can grow and shrink at runtime by allocating and deallocating memeory. So there is no need to
give initial size of linked list.

- - Insertion and Deletion

Insertion and deletion of nodes are really easier. Unlike array here we don’t have to shift elements after insertion or deletion of an element.
In linked list we just have to update the address present in next pointer of a node.

- No Memory Wastage

As size of linked list can increase or decrease at run time so there is no memory wastage. In case of array there is lot of memory wastage,
like if we declare an array of size 10 and store only 6 elements in it then space of 4 elements are wasted. There is no such problem in linked list
as memory is allocated only when required.

- Implementation
Data structures such as stack and queues can be easily implemented using linked list.

- Disadvantages of Linked List
- Memory Usage

More memory is required to store elements in linked list as compared to array. Because in linked list each node cont ains a pointer and it
requires extra memory for itself.
- Traversal

Elements or nodes traversal is difficult in linked list. We can not randomly access any element as we do in array by i ndex. For example if
we want to access a node at position n then we have to traverse all the nodes before it. So, time required to access a n ode is large.

- Reverse Traversing

In linked list reverse traversing is really difficult. In case of doubly linked list its easier but extra memory is required for back pointer
hence wastage of memory.

Q6. Mention what is traversal in linked lists?

Term Traversal is used to refer the operation of processing each element in the list.

Q7.
Mention what is the difference between Linear Array and Linked List?

The difference between Linear Array and Linked List are shown below,

Linear Array Linked List
Deletion and Insertions are difficult.
Deletion and Insertions can be done easily.
For insertion and deletion, it needs movements
For insertion and deletion, it does not require movement of nodes
In it space is wasted
In it space is not wasted
It is expensive
It is not expensive
It cannot be reduced or extended according to requirements
It can be reduced or extended according to requirements
To avail each element same amount of time is required.
To avail each element different amount of time is required.
In consecutive memory locations elements are stored.
Elements may or may not be stored in consecutive memory locations

We can reach there directly if we have to go to a particular element
To reach a particular node, you need to go through all those nodes that come before that node.

https://career.guru99.com/top-17-linked-list-interview-questions/

7) Mention what are the applications of Linked Lists?

Applications of Linked Lists are,

Linked lists are used to implement queues, stacks, graphs, etc.
In Linked Lists you don't need to know the size in advance.
Linked lists let you insert elements at the beginning and end of the list.
8) What does the dummy header in linked list contain?

In linked list, the dummy header contains the first record of the actual data

9) Mention the steps to insert data at the starting of a singly linked list?

Steps to insert data at the starting of a singly linked list include,

Create a new node
Insert new node by allocating the head pointer to the new node next pointer
Updating the head pointer to the point the new node.

11) Mention what are the applications that use Linked lists?

Both queues and stacks are often implemented using linked lists. Other applications are list, binary tree, skip, unroll
ed linked list, hash
table, etc.

12) Explain how to add an item to the beginning of the list?

To add an item to the beginning of the list, you have to do the following:

Create a new item and set its value
Link the new item to point to the head of the list
Set the head of the list to be our new item
If you are using a function to do this operation, you need to alter the head variable. To do this, you must pass a point
er to the pointer
variable (a double pointer). so you will be able to modify the pointer itself.

3) Mention what is the biggest advantage of linked lists?

The biggest benefit of linked lists is that you do not specify a fixed size for your list. The more elements you add to t
he chain, the bigger
the chain gets.

14) Mention how to delete first node from singly linked list?

To delete first node from singly linked list

Store Current Start in Another Temporary Pointer

Move Start Pointer One position Ahead
Delete temp i.e Previous Starting Node as we have Updated Version of Start Pointer
15) Mention how to display Singly Linked List from First to Last?

To display Singly Linked List from First to Last,

Create a linked list using create().
You cannot change the address stored inside global variable “start” therefore you have to declare one temporary vari able -“temp” of type node
To traverse from start to end, you should allot address of Starting node in Pointer variable i.e temp.

16) Mention how to insert a new node in linked list where free node will be available?

To insert a new node in linked list the free node will be available in Avail list.

17) Mention for which header list, you will found the last node contains the null pointer?

For grounded header list you will found the last node contains the null pointer.