1.Which of the following sorting algorithms can be used to sort a random linked list with minimum time complexity?  
**(A)** Insertion Sort  
**(B)** Quick Sort  
**(C)** Heap Sort  
**(D) Merge Sort**

2.Which of the following points is/are true about Linked List data structure when it is compared with array  
**(A)** Arrays have better cache locality that can make them better in terms of performance.  
**(B)** It is easy to insert and delete elements in Linked List  
**(C)** Random access is not allowed in a typical implementation of Linked Lists  
**(D)** The size of array has to be pre-decided, linked lists can change their size any time.  
**(E) All of the above**

3.In the worst case, the number of comparisons needed to search a singly linked list of length n for a given element is

**(A)** log 2 n  
**(B)** n/2  
**(C)** log 2 n – 1  
**(D) n**

4.Suppose each set is represented as a linked list with elements in arbitrary order. Which of the operations among union, intersection, membership, cardinality will be the slowest? **(A)** union only  
**(B)** intersection, membership  
**(C)** membership, cardinality  
**(D) union, intersection**

5. Which of the following is not a disadvantage to the usage of array?  
a) Fixed size  
b) There are chances of wastage of memory space if elements inserted in an array are lesser than the allocated size  
c) Insertion based on position  
**d) Accessing elements at specified positions**

6.Explanation: Array elements can be accessed in two steps. First, multiply the size of the data type with the specified position, second, add this value to the base address. Both of these operations can be done in constant time, hence accessing elements at a given index/position is faster.

7.What is the time complexity of inserting at the end in dynamic arrays?  
a) O(1)  
b) O(n)  
c) O(logn)  
**d) Either O(1) or O(n)**

8.Explanation: Depending on whether the array is full or not, the complexity in dynamic array varies. If you try to insert into an array that is not full, then the element is simply stored at the end, this takes O(1) time. If you try to insert into an array which is full, first you will have to allocate an array with double the size of the current array and then copy all the elements into it and finally insert the new element, this takes O(n) time.

9.What is the time complexity to count the number of elements in the linked list?  
a) O(1)  
**b) O(n)**  
c) O(logn)  
d) O(n2)

Explanation: To count the number of elements, you have to traverse through the entire list, hence complexity is O(n).

**10 In a circular linked list**  
  
a) Components are all linked together in some sequential manner.  
**b) There is no beginning and no end.**c) Components are arranged hierarchically.  
d) Forward and backward traversal within the list is permitted.

**11. A linear collection of data elements where the linear node is given by means of pointer is called?**  
  
**a) Linked list**b) Node list  
c) Primitive list  
d) None

**12.Which of the following operations is performed more efficiently by doubly linked list than by singly linked list?**  
  
**a) Deleting a node whose location in given**  
b) Searching of an unsorted list for a given item  
c) Inverting a node after the node with given location  
d) Traversing a list to process each node

**A variant of linked list in which last node of the list points to the first node of the list is?**  
  
a) Singly linked list  
b) Doubly linked list  
**c) Circular linked list**d) Multiply linked list

**In doubly linked lists, traversal can be performed?**  
  
a) Only in forward direction  
b) Only in reverse direction  
**c) In both directions**d) None

**What kind of linked list is best to answer question like “What is the item at position n?”**  
  
a) Singly linked list  
b) Doubly linked list  
c) Circular linked list  
**d) Array implementation of linked list**

What does the following function do for a given Linked List with first node as head?

void fun1(struct node\* head)

{

if(head == NULL)

return;

fun1(head->next);

printf("%d ", head->data);

}

A. Prints all nodes of linked lists  
**B. Prints all nodes of linked list in reverse order**C. Prints alternate nodes of Linked List  
D. Prints alternate nodes in reverse order

Explanation: fun1() prints the given Linked List in reverse manner. For Linked List 1->2->3->4->5, fun1() prints 5->4->3->2->1.

 A linear collection of data elements where the linear node is given by means of pointer is called?

**A. linked list**B. node list  
C. primitive list  
D. None of these

Which of these is an application of linked lists?

A. To implement file systems  
B. For separate chaining in hash-tables  
C. To implement non-binary trees  
**D. All of the mentioned**

Explanation: Linked lists can be used to implement all of the above mentioned applications.

10. In circular linked list, insertion of node requires modification of?

A. One pointer  
**B. Two pointer**C. Three pointer  
D. None

Explanation: In circular linked list, insertion of node requires modification of Two pointer.

Generally collection of Nodes is called as \_\_\_\_\_\_\_\_\_\_.

|  |
| --- |
| 1.Heap  2.Pointer  **3.Linked List**  4.Stack |
|  |

A linear collection of data element given by mean of pointer is called \_\_\_\_\_\_\_\_\_\_\_\_\_\_.

|  |
| --- |
| 1.Queue  2.**Linked List**  3.Graph  4.Stack |

Which of the following is not a type of Linked List ?

|  |
| --- |
| **1.Hybrid Linked List**  2.Singly Linked List  3.Doubly Linked List  4.Hybrid Linked List |

Linked list is generally considered as an example of \_\_\_\_\_\_\_\_\_ type of memory allocation.

|  |
| --- |
| 1.Static  2.Compile Time  **3.Dynamic**  4.None of these |

Each Node contain minimum two fields one field called data field to store data. Another field is of type \_\_\_\_\_\_\_\_\_.

1.Pointer to Character

2.Pointer to an Integer

3.Pointer to Class

**4.Pointer to Node**

**8)**Which linked lists do not have ends?

a) Single linked lust

b) doubly linked list

**c) circular linked list**

d) none

A non-circular doubly linked list can best and most generally be defined as a \_\_\_

(A) Set of elements, each with two pointers

(B) Set of elements chained together with pointers

(C) Linear sequence of elements in sequential memory locations

**(D) Linear sequence of elements chained together with pointers**

Which of the following operations is a dictionary operation?

(A) Search

(B) Delete

(C) Insert

**(D) All of above**

To create a linked structure, each node must have one member, which is \_\_\_\_

(A) A pointer to the head of the list

(B) A pointer to NULL

**(C) A pointer to the node type**

(D) A reference to the element type

The situation when in a linked list START=NULL is

**1.Underflow**

2.overflow

3.housefull

4.saturated

What would be the asymptotic time complexity to add a node at the end of singly linked list, if the pointer is initially pointing to the head of the list?

1.O(1)

2.O(n)

**3.θ (n)**

4.θ (1)