Array and Linked List Data Structure
Assignment -#1

1. Which of these best describes an array?
   a) A data structure that shows a hierarchical behavior
   b) Container of objects of similar types
   c) Arrays are immutable once initialized
   d) More than one of the above
   e) None of the above

2. How do you initialize an array in C?
   a) int arr[3] = (1,2,3);
   b) int arr(3) = {1,2,3};
   c) int arr[3] = {1,2,3};
   d) More than one of the above
   e) None of the above

3. How do you instantiate an array in Java?
   a) int arr[] = new int(3);
   b) int arr[];
   c) int arr[] = new int[3];
   d) More than one of the above
   e) None of the above

4. Which of the following is the correct way to declare a multidimensional array in Java?
   a) int[] arr;
   b) int arr[[]];
   c) int[][]arr;
   d) More than one of the above
   e) None of the above

5. What is the output of the following Java code?

```
public class array
{
   public static void main(String args[])
   {
      int []arr = {1,2,3,4,5};
      System.out.println(arr[2]);
      System.out.println(arr[4]);
   }
}
```

   a) 3 and 5
   b) 5 and 3
   c) 2 and 4
   d) More than one of the above
   e) None of the above

6. What is the output of the following Java code?

```
public class array
{
   public static void main(String args[])
   {
      int []arr = {1,2,3,4,5};
      System.out.println(arr[5]);
   }
}
```

   a) 4
   b) 5
   c) ArrayIndexOutOfBoundsException
   d) More than one of the above
   e) None of the above

7. When does the ArrayIndexOutOfBoundsException occur?
   a) Compile-time
   b) Run-time
   c) Not an error
   d) More than one of the above
   e) None of the above

8. Which of the following concepts make extensive use of arrays?
   a) Binary trees
   b) Scheduling of processes
   c) Caching
   d) Spatial locality
   e) More than one of the above

9. What are the advantages of arrays?
   a) Objects of mixed data types can be stored
   b) Elements in an array cannot be sorted
   c) Index of the first element of an array is 1
   d) Easier to store elements of the same data type
   e) More than one of the above

10. What are the disadvantages of arrays?
   a) Data structures like queue or stack cannot be implemented
   b) There are chances of wastage of memory space if elements inserted in an array are lesser than the allocated size
   c) Index value of an array can be negative
   d) Elements are sequentially accessed
   e) More than one of the above

11. Assuming int is of 4 bytes, what is the size of int arr[15];?
    a) 45
    b) 19
    c) 30
    d) 60
    e) None of the above

12. In general, the index of the first element in an array is _____.
    a) 0
    b) -1
    c) 2
    d) More than one of the above
    e) None of the above

13. Elements in an array are accessed _____.
    a) Randomly
    b) Sequentially
    c) Exponentially
    d) Logarithmically
    e) More than one of the above

14. Which of the following is the limitation of the array?
    a) Elements can be accessed from anywhere
    b) The size of the array is fixed
    c) Indexing is started from Zero
    d) Memory waste if an array's elements are smaller than the size allotted to them
    e) More than one of the above

15. In an array int arr[3]={1,2,3}, what will happen if we try to access arr[4] in C/C++?
    a) Run-time error
    b) 3
    c) 0
    d) Garbage Value
    e) More than one of the above

16. What is the time complexity to insert a single element in the array at the end?
    a) O(1)
    b) O(n)
    c) O(log n)
    d) More than one of the above
    e) None of the above

17. If we declare a 2D array like int arr[3][4], then it will be stored in the format of _____.
    a) 4 rows 3 columns
    b) 4 rows 4 columns
    c) 3 rows 3 columns
    d) 3 rows 4 columns
    e) None of the above

18. What is the time complexity for traversing a 2D array?
    a) O(n)
    b) O(nlogn)
    c) $O(n^2)$
    d) More than one of the above
    e) None of the above

19. What is the output of the following program?

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
   int arr[] = { 1, 2, 3, 4, 5 };
   for (int i = 0; i < 5; i++)
      cout << *(arr + i) << endl;
   return 0;
}
```

    a) 1, 3, 5, 7, 9
    b) 1200, 1201, 1202, 1203, 1204
    c) 1, 2, 3, 4, 5
    d) More than one of the above
    e) None of the above

20. What is the output of the following program?

```
#include <iostream>
using namespace std;
int main()
{
   int val = 10;
   int* ptr = &val;
   cout << *ptr << endl;
   cout << ptr << endl;
   return 0;
}
```

    a) Address of val, Address of val
    b) Address of val, 10
    c) 10, 10
    d) More than one of the above
    e) None of the above

21. What is the time complexity for inserting/deleting at the beginning of the array?
    a) O(1)
    b) O(logN)
    c) O(NlogN)
    d) O(N)
    e) None of the above

22. In C++, we can create a dynamic array using the _____ keyword.
    a) array
    b) this
    c) super
    d) new
    e) None of the above

23. What is the advantage of a dynamic array over a static array?
    a) The dynamic array is fixed
    b) The dynamic array takes O(n) time
    c) The size of the dynamic array is not fixed
    d) More than one of the above
    e) None of the above

24. How can we create a dynamic array in C?
    a) vector()
    b) dynamic_array()
    c) malloc()
    d) More than one of the above
    e) None of the above

25. Consider a 2-dimensional array a[4][5], the array is stored in row-major format. If the first element x[0][0] occupies the memory location with address 1000 and each element occupies only 4 bytes, find the address of the element A[2][3].
    a) 1048
    b) 1052
    c) 1060
    d) 1072
    e) None of the above

26. 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) $O(n^2)$
    e) More than one of the above

27. What is the time complexity for deleting a node from linked list?
    a) O(1)
    b) O(n)
    c) O(logn)
    d) More than one of the above
    e) None of the above

28. Which of these is not an application of a linked list?
    a) To implement file systems
    b) For separate chaining in hash-tables
    c) To implement non-binary trees
    d) Random Access of elements
    e) More than one of the above

29. A linear collection of data elements where the linear node is given by a pointer is called?
    a) Linked list
    b) Node list
    c) Primitive list
    d) Unordered list
    e) More than one of the above

30. Consider an implementation of unsorted singly linked list. Suppose it has its representation with a head pointer only. Given the representation, which of the following operation can be implemented in O(1) time?
    i) Insertion at the front of the linked list
    ii) Insertion at the end of the linked list
    iii) Deletion of the front node of the linked list
    iv) Deletion of the last node of the linked list
    a) I and II
    b) I and III
    c) I, II and III
    d) I, II and IV
    e) More than one of the above

31. In linked list, each node contains a minimum of two fields. One field is the data field to store the data. The second field is?
    a) Pointer to character
    b) Pointer to integer
    c) Pointer to node
    d) Node
    e) More than one of the above

32. 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?
    a) O(1)
    b) O(n)
    c) $\theta(n^2)$
    d) More than one of the above
    e) None of the above

33. What would be the asymptotic time complexity to insert an element at the front of the linked list (head is known)?
    a) O(1)
    b) O(n)
    c) $O(n^2)$
    d) More than one of the above
    e) None of the above

34. What would be the asymptotic time complexity to find an element in the linked list?
    a) O(1)
    b) O(n)
    c) $O(n^2)$
    d) More than one of the above
    e) None of the above

35. What would be the asymptotic time complexity to insert an element at the second position in the linked list?
    a) O(1)
    b) O(n)
    c) $O(n^2)$
    d) More than one of the above
    e) None of the above

36. The concatenation of two lists can be performed in O(1) time. Which of the following variation of the linked list can be used?
    a) Singly linked list
    b) Doubly linked list
    c) Circular doubly linked list
    d) More than one of the above
    e) None of the above

37. Which data structure is typically used to implement hash table ?
    a) Linked list
    b) Array
    c) Binary Tree
    d) More than one of the above
    e) None of the above

38. What kind of linked list is best to answer questions 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
    e) More than one of the above

39. Linked lists are not suitable for the implementation of _____.
    a) Insertion sort
    b) Radix sort
    c) Polynomial manipulation
    d) Binary search
    e) More than one of the above

40. Linked list is considered as an example of _____ type of memory allocation.
    a) Dynamic
    b) Static
    c) Compile time
    d) Heap
    e) More than one of the above

41. In Linked List implementation, a node carries information regarding _____.
    a) Data
    b) Link
    c) Data and Link
    d) Node
    e) More than one of the above

42. Linked list data structure offers considerable saving in _____.
    a) Computational Time
    b) Space Utilization
    c) Speed Utilization
    d) More than one of the above
    e) None of the above

43. Which of the following points is/are not true about Linked List data structure when it is compared with an 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) Access of elements in linked list takes less time than compared to arrays
    e) More than one of the above

44. 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

45. 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) More than one of the above
e) None of the above

46. Given a pointer to a node X in a singly linked list. Only one pointer is given, pointer to the head node is not given. Can we delete the node X from the given linked list?
a) Possible if X is not the last node
b) Possible if the size of the linked list is even
c) Possible if the size of the linked list is odd
d) Possible if X is not the first node
e) More than one of the above

47. You are given pointers to the first and last nodes of a singly linked list. Which of the following operations are dependent on the length of the linked list?
a) Delete the first element
b) Insert a new element as the first element
c) Delete the last element of the list
d) Add a new element at the end of the list
e) More than one of the above

48. 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) More than one of the above
e) None of the above

49. Which of the following is false about a doubly linked list?
a) We can navigate in both directions
b) It requires more space than a singly linked list
c) The insertion and deletion of a node take a bit longer
d) Implementing a doubly linked list is easier than a singly linked list
e) More than one of the above

50. What is a memory-efficient double linked list?
a) Each node has only one pointer to traverse the list back and forth
b) The list has breakpoints for faster traversal
c) An auxiliary singly linked list acts as a helper list to traverse through the doubly linked list
d) A doubly linked list that uses a bitwise AND operator for storing addresses
e) More than one of the above

51. How do you calculate the pointer difference in a memory-efficient doubly linked list?
a) Head xor tail
b) Pointer to previous node xor pointer to next node
c) Pointer to previous node – pointer to next node
d) Pointer to next node – pointer to previous node
e) None of the above

52. What is the worst-case time complexity of inserting a node in a doubly linked list?
a) O(nlogn)
b) O(logn)
c) O(n)
d) More than one of the above
e) None of the above

53. What differentiates a circular linked list from a normal linked list?
a) You cannot have the 'next' pointer point to null in a circular linked list
b) It is faster to traverse the circular linked list
c) In a circular linked list, each node points to the previous node instead of the next node
d) The head node is known in a circular linked list
e) More than one of the above

54. What is the time complexity of searching for an element in a circular linked list?
a) O(n)
b) O(nlogn)
c) O(1)
d) $O(n^2)$

55. Which of the following applications makes use of a circular linked list?
    a) Undo operation in a text editor
    b) Recursive function calls
    c) Allocating CPU to resources
    d) Implementing hash tables
    e) More than one of the above

56. Which of the following is false about a circular linked list?
    a) Every node has a successor
    b) Time complexity of inserting a new node at the head of the list is O(1)
    c) Time complexity for deleting the last node is O(n)
    d) We can traverse the whole circular linked list by starting from any point
    e) More than one of the above

57. Consider a small circular linked list. How can you effectively detect the presence of cycles in this list?
    a) Keep one node as head and traverse another temp node until the end to check if its 'next' points to head
    b) Have fast and slow pointers, with the fast pointer advancing two nodes at a time and the slow pointer advancing by one node at a time
    c) Cannot determine, you have to pre-define if the list contains cycles
    d) A circular linked list itself represents a cycle, so no new cycles can be generated
    e) More than one of the above

58. Which of the following methods can be used to search for an element in a linked list?
    a) Iterative linear search
    b) Iterative binary search
    c) Recursive binary search
    d) Normal binary search
    e) More than one of the above

59. What will be the time complexity when binary search is applied to a linked list?
    a) O(1)
    b) O(n)
    c) $O(n^2)$
    d) More than one of the above
    e) None of the above

60. Which type of linked list stores the address of the header node in the next field of the last node?
    a) Singly linked list
    b) Circular linked list
    c) Doubly linked list
    d) Hashed list
    e) None of the above

Solution with Explanation

Answer1: b Container of objects of similar types
Explanation: Array contains elements only of the same type.

Answer2: c int arr[3] = {1,2,3};
Explanation: This is the syntax to initialize an array in C.

Answer3: c int arr[] = new int[3];
Explanation: Note that int arr[]; is declaration whereas int arr[] = new int[3]; is to instantiate an array.

Answer4: c int[][]arr;
Explanation: The syntax to declare multidimensional array in java is either int[][] arr; or int arr[][];

Answer5: a) 3 and 5
Explanation: Array indexing starts from 0.

Answer6: c ArrayIndexOutOfBoundsException
Explanation: Trying to access an element beyond the limits of an array gives ArrayIndexOutOfBoundsException.

Answer7: b Run-time
Explanation: ArrayIndexOutOfBoundsException is a run-time exception and the compilation is error-free.

Answer8: d Spatial locality
Explanation: Whenever a particular memory location is referred to, it is likely that the locations nearby are also referred, arrays are stored as contiguous blocks in memory, so if you want to access array elements, spatial locality makes it to access quickly.

Answer9: d) Easier to store elements of same data type
Explanation: Arrays store elements of the same data type and present in continuous memory locations.

Answer10: b) There are chances of wastage of memory space if elements inserted in an array are lesser than the allocated size
Explanation: Arrays are of fixed size. If we insert elements less than the allocated size, unoccupied positions can't be used again. Wastage will occur in memory.

Answer11: d) 60
Explanation: Since there are 15 int elements and each int is of 4bytes, we get 15*4 = 60bytes.

Answer12: a) 0
Explanation: In general, Array Indexing starts from 0. Thus, the index of the first element in an array is 0.

Answer13: a) randomly
Explanation: Elements in an array are accessed randomly. In Linked lists, elements are accessed sequentially.

Answer 14: d) Memory waste if an array's elements are smaller than the size allotted to them
Suppose we have an arry like : arr[5] ={1, 2, 3}
Here we have declared the size of the array as 5, but we have initialized only 3 values to it. So it leads to memory wastage.

Answer15: d) Garbage Value
If you try to access arr[4], which is beyond the valid range of indices, it will result in undefined behavior. as providing garbage values.

Answer16: a) O(1)
The time Complexity to insert a single element in the array is O(1). Because we can access it from anywhere(means from start, from middle, from ending, or from anywhere) with the help of indices.

Answer17: d) 3 rows 4 coloumn
If we declare an array as - int arr[3][4] that means 3 rows and 4 columns. The first subscript will be the number of rows and the second subscript will be the number of columns.

Answer18: c) O(n^2)
For traversing a 2d array there are two loops(nested loops) required One for the rows and the other for the columns. As we know that a loop will take O(n) time complexity, and the nested loop will take O(n^2) time complexity. So the time complexity for traversing a 2d array is O(n^2).

Answer19: c) 1, 2, 3, 4, 5
arr[i] is interpreted as *(arr + i). Now, arr is the address of the array or address of 0th index element of the array. So, the address of next element in the array is arr + 1 (because elements in the array are stored in consecutive memory locations), the further address of the next location is arr + 2, and so on. Going with the above arguments, (arr + i) means the address at i distance away from the starting element of the array.
Therefore, it is another method for traversing the array.

Answer20: e) 10, address of val
In the above code,
int val = 10;
int* ptr = &val;
Here ptr means the variable which is storing the address of variable val, and *ptr means the variable located at the address specified by its operand.
So *ptr = 10.
and ptr =address of val.

Answer21: d) O(n)
Insert operation in an array at any position can be performed by shifting elements to the right, which are on the right side of the required position. so it will take O(N) time.

Answer22:d) New
In C++, you can create a dynamic array using the new keyword. The new keyword is used to dynamically allocate memory for an array on the heap during runtime. It allows you to create an array of a specified size and returns a pointer to the first element of the allocated memory.
int* dynamicArray = new int[size];

Answer23: c) The size of the dynamic array is not fixed.
A Dynamic Array is allocated memory at runtime and its size can be changed later in the program. the size of a dynamic array is not fixed. it means you can increase or decrease its size as needed, allowing for more flexibility in handling varying amounts of data. This dynamic resizing is particularly useful when you don't know the exact size of the data beforehand.

Answer24: c) malloc()
A Dynamic Array is allocated memory at runtime and its size can be changed later in the program.
We can create a dynamic array in C by using the following methods:
Using malloc() Function
Using calloc() Function
Resizing Array Using realloc() Function
Using Variable Length Arrays(VLAs)
Using Flexible Array Members

Answer25: b) 1052
In a 2D array stored in Row Major Order, the elements are stored row by row. To calculate the address of any element A[i][j], you use the following formula:
Address of A[i][j]=B+[i×C+j]×W
Address of A[2][3]=1000+[2×5+3]×4 =1052

Answer26: b O(n)
Explanation: To count the number of elements, you have to traverse through the entire list, hence complexity is O(n).

Answer27: a O(n)
Explanation: You need to traverse till the node, hence the time complexity is O(n).

Answer28: d Random Access of elements
Explanation: To implement file system, for separate chaining in hash-tables and to implement non-binary trees linked lists are used. Elements are accessed sequentially in linked list. Random access of elements is not an applications of linked list.

Answer29: a Linked list
Explanation: In Linked list each node has its own data and the address of next node. These nodes are linked by using pointers. Node list is an object that consists of a list of all nodes in a document with in a particular selected set of nodes.

Answer30: b) I and III
Explanation: We know the head node in the given linked list. Insertion and deletion of elements at the front of the linked list completes in O (1) time whereas for insertion and deletion at the last node requires to traverse through every node in the linked list. Suppose there are n elements in a linked list, we need to traverse through each node. Hence time complexity becomes O(n).

Answer31: c Pointer to node
Explanation: Each node in a linked list contains data and a pointer (reference) to the next node. Second field contains pointer to node.

Answer32: b) O(n)
Explanation: In case of a linked list having n elements, we need to travel through every node of the list to add the element at the end of the list. Thus asymptotic time complexity is both θ(n) and O(n). Θ(n) represents the tight bound of the algorithm's time complexity, meaning it captures the best, average, and worst-case scenarios that are all linear in this case. O(n) signifies the upper bound, indicating the worst-case scenario is no worse than linear.

Answer33: a O(1)
Explanation: To add an element at the front of the linked list, we will create a new node which holds the data to be added to the linked list and pointer which points to head position in the linked list. The entire thing happens within O (1) time. Thus the asymptotic time complexity is O (1).

Answer34: b O(n)
Explanation: If the required element is in the last position, we need to traverse the entire linked list. This will take O (n) time to search the element.

Answer35: a O(1)
Explanation: A new node is created with the required element. The pointer of the new node points the node to which the head node of the linked list is also pointing. The head node pointer is changed and it points to the new node which we created earlier. The entire process completes in O (1) time. Thus the asymptotic time complexity to insert an element in the second position of the linked list is O (1).

Answer36: c Circular doubly linked list
Explanation: We can easily concatenate two lists in O (1) time using singly or doubly linked list, provided that we have a pointer to the last node at least one of the lists. But in case of circular doubly linked lists, we will break the link in both the lists and hook them together. Thus circular doubly linked list concatenates two lists in O (1) time.

Answer37: a) linked list
Explanation: hash table implemented by using lined list

Answer38: d Array implementation of linked list
Explanation: Arrays provide random access to elements by providing the index value within square brackets. In the linked list, we need to traverse through each element until we reach the nth position. Time taken to access an element represented in arrays is less than the singly, doubly and circular linked lists. Thus, array implementation is used to access the item at the position n.

Answer39: d Binary search
Explanation: It cannot be implemented using linked lists.

Answer40: a Dynamic
Explanation: As memory is allocated at the run time.

Answer41: c Data and Link
Explanation: A linked list is a collection of objects linked together by references from an object to another object. By convention these objects are names as nodes. Linked list consists of nodes where each node contains one or more data fields and a reference(link) to the next node.

Answer42: d) Space Utilization and Computational Time
Explanation: Linked lists saves both space and time.

Answer43: d Access of elements in linked list takes less time than compared to arrays
Explanation: To access an element in a linked list, we need to traverse every element until we reach the desired element. This will take more time than arrays as arrays provide random access to its elements.

Answer44: d Merge Sort
Explanation: Both Merge sort and Insertion sort can be used for linked lists. The slow random-access performance of a linked list makes other algorithms (such as quicksort) perform poorly, and others (such as heapsort) completely impossible. Since worst case time complexity of Merge Sort is O(nLogn) and Insertion sort is $O(n^2)$, merge sort is preferred.

Answer45: e O(n)
Explanation: In the worst case, the element to be searched has to be compared with all elements of the linked list.

Answer46: a Possible if X is not last node
Explanation: Following are simple steps.

```
struct node *temp  = X->next;
X->data  = temp->data;
X->next  = temp->next;
free(temp);
```

Answer47: c) Delete the last element of the list
Explanation: Deletion of the first element of the list is done in O (1) time by deleting memory and changing the first pointer. Insertion of an element as a first element can be done in O (1) time. We will create a node that holds data and points to the head of the given linked list. The head pointer was changed to a newly created node. Deletion of the last element requires a pointer to the previous node of last, which can only be obtained by traversing the list. This requires the length of the linked list.Adding a new element at the end of the list can be done in O (1) by changing the pointer of the last node to the newly created node and last is changed to a newly created node.

Answer48: d) n
Explanation: The worst-case happens if the required element is at last or the element is absent in the list. For this, we need to compare every element in the linked list. If n elements are there, n comparisons will happen in the worst case.

Answer49: d) Implementing a doubly linked list is easier than singly linked list
Explanation: A doubly linked list has two pointers 'left' and 'right' which enable it to traverse in either direction. Compared to singly liked list which has only a 'next' pointer, doubly linked list requires extra space to store this extra pointer. Every insertion and deletion requires manipulation of two pointers, hence it takes a bit longer time. Implementing doubly linked list involves setting both left and right pointers to correct nodes and takes more time than singly linked list.

Answer50: a Each node has only one pointer to traverse the list back and forth
Explanation: Memory efficient doubly linked list has only one pointer to traverse the list back and forth. The implementation is based on pointer difference. It uses bitwise XOR operator to store the front and rear pointer addresses. Instead of storing actual memory address, every node store the XOR address of previous and next nodes.

Answer51: b pointer to previous node xor pointer to next node
Explanation: The pointer difference is calculated by taking XOR of pointer to previous node and pointer to the next node.

Answer52: c O(n)
Explanation: In the worst case, the position to be inserted maybe at the end of the list, hence you have to traverse through the entire list to get to the correct position, hence O(n).

Answer53: a You cannot have the 'next' pointer point to null in a circular linked list
Explanation: In a normal linked list, the 'next' pointer of the last node points to null. However, in a circular linked list, the 'next' pointer of the last node points to the head (first element) of the list. Every node in a circular linked list can be a starting point(head).

Answer54: a O(n)
Explanation: In the worst case, you have to traverse through the entire list of n elements.

Answer55: c Allocating CPU to resources
Explanation: Generally, round robin fashion is employed to allocate CPU time to resources which makes use of the circular linked list data structure. Recursive function calls use stack data structure. Undo Operation in text editor uses doubly linked lists. Hash tables uses singly linked lists.

Answer56: b) Time complexity of inserting a new node at the head of the list is O(1)
Explanation: Time complexity of inserting a new node at the head of the list is O(n) because you have to traverse through the list to find the tail node.

Answer57: b Have fast and slow pointers with the fast pointer advancing two nodes at a time and slow pointer advancing by one node at a time
Explanation: Advance the pointers in such a way that the fast pointer advances two nodes at a time and slow pointer advances one node at a time and check to see if at any given instant of time if the fast pointer points to slow pointer or if the fast pointer's 'next' points to the slow pointer. This is applicable for smaller lists.

Answer58: a Iterative linear search
Explanation: Iterative linear search can be used to search an element in a linked list. Binary search can be used only when the list is sorted.

Answer59: b O(n)
Explanation: The time complexity will be O(n) when binary search is applied on a linked list.

Answer 60: c) Circular linked list
Explanation: last node connected with first node