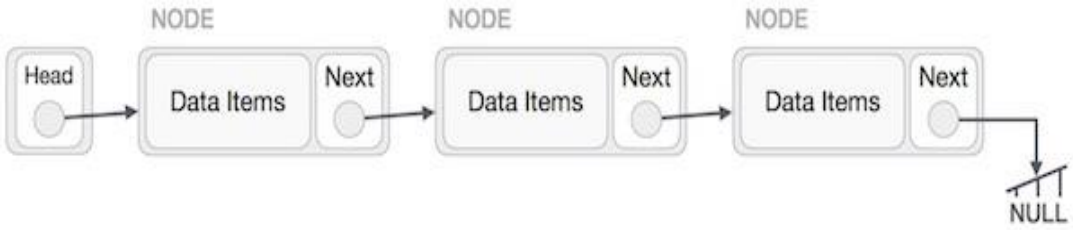
DATASTRUCTURE MATERIALS FOR MCA ENTRANCE EXAMINATION

LINKED LIST BASIC

A linked list is a sequence of data structures, which are connected together via links. Linked List is a sequence of links which contains items. Each link contains a connection to another link. Linked list is the second most-used data structure after array. Following are the important terms to understand the concept of Linked List.

- **Link** − Each link of a linked list can store a data called an element.
- **Next** − Each link of a linked list contains a link to the next link called Next.
- **LinkedList** − A Linked List contains the connection link to the first link called First

Linked List Representation



IMPORTANT POINTS IN LINKED LIST

- Linked List contains a link element called first.
- Each link carries a data field(s) and a link field called next.
- Each link is linked with its next link using its next link.
- Last link carries a link as null to mark the end of the list.

TYPES OF LINKEDLIST

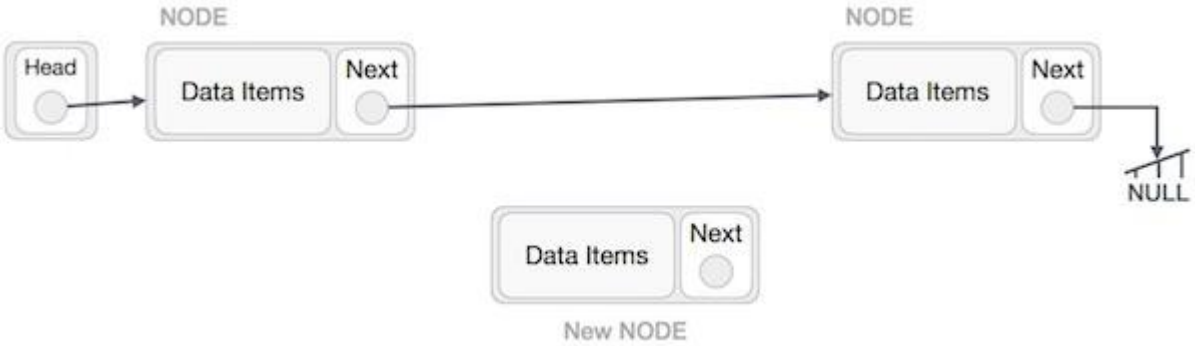
- **Simple Linked List** − Item navigation is forward only.
- **Doubly Linked List** − Items can be navigated forward and backward.
- **Circular Linked List** − Last item contains link of the first element as next and the first element has a link to the last element as previous.

BASICS OPERATIONS

- **Insertion** − Adds an element at the beginning of the list.
- **Deletion** − Deletes an element at the beginning of the list.
- **Display** − Displays the complete list.
- **Search** − Searches an element using the given key.
- **Delete** − Deletes an element using the given key.

Insertion Operation

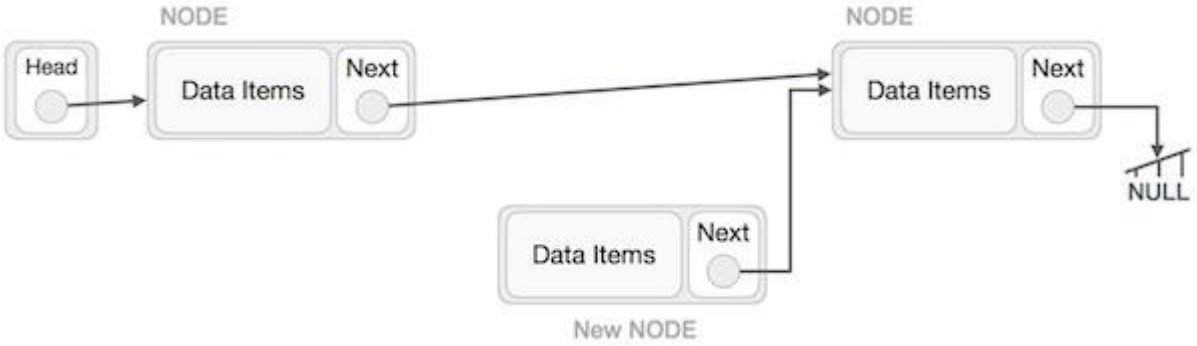
Adding a new node in linked list is a more than one step activity. We shall learn this with diagrams here. First, create a node using the same structure and find the location where it has to be inserted.



Imagine that we are inserting a node **B** (NewNode), between **A** (LeftNode) and **C** (RightNode). Then point B.next to C −

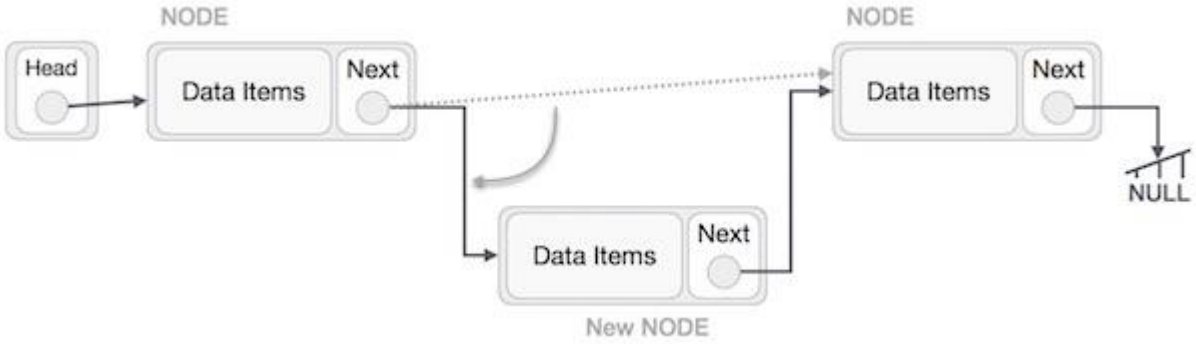
```
NewNode.next -> RightNode;
```

It should look like this −



Now, the next node at the left should point to the new node.

```
LeftNode.next -> NewNode;
```



This will put the new node in the middle of the two. The new list should look like this –

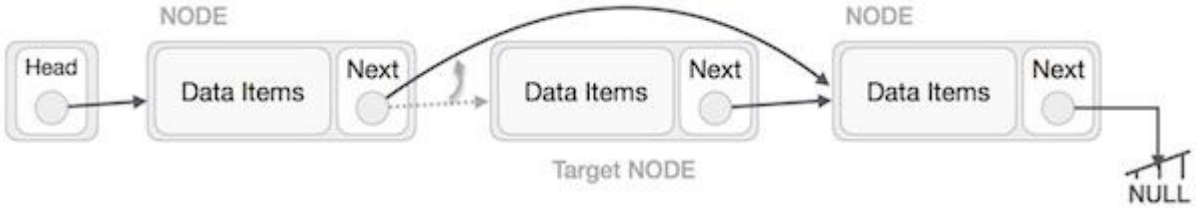
Similar steps should be taken if the node is being inserted at the beginning of the list. While inserting it at the end, the second last node of the list should point to the new node and the new node will point to NULL.

Deletion Operation

Deletion is also a more than one step process. We shall learn with pictorial representation. First, locate the target node to be removed, by using searching algorithms.

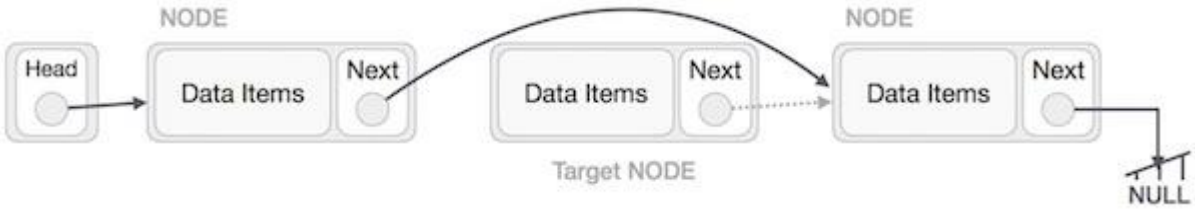
The left (previous) node of the target node now should point to the next node of the target node −

```
LeftNode.next -> TargetNode.next;
```

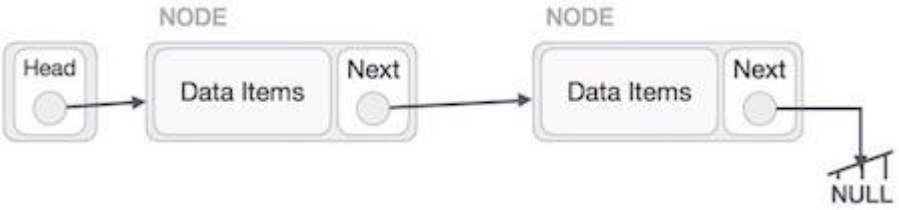


This will remove the link that was pointing to the target node. Now, using the following code, we will remove what the target node is pointing at.

```
TargetNode.next -> NULL;
```

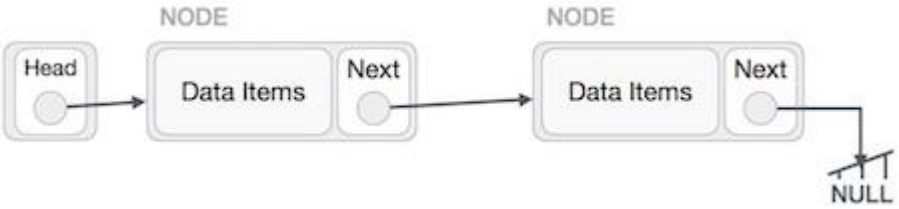


We need to use the deleted node. We can keep that in memory otherwise we can simply deallocate memory and wipe off the target node completely.

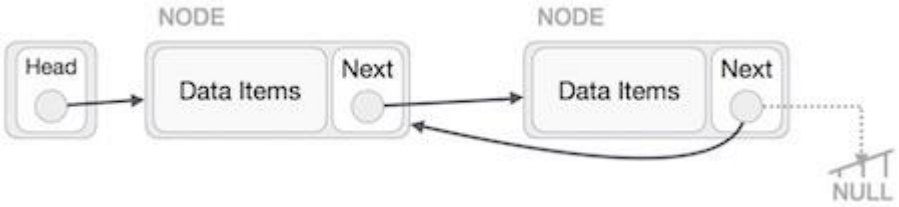


Reverse Operation

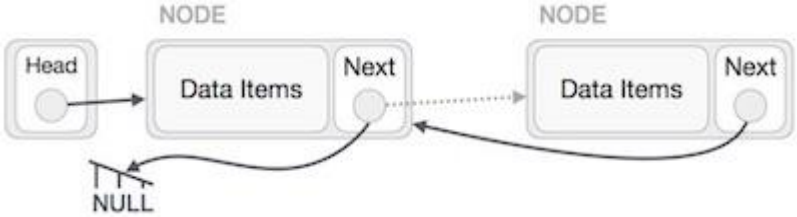
This operation is a thorough one. We need to make the last node to be pointed by the head node and reverse the whole linked list.



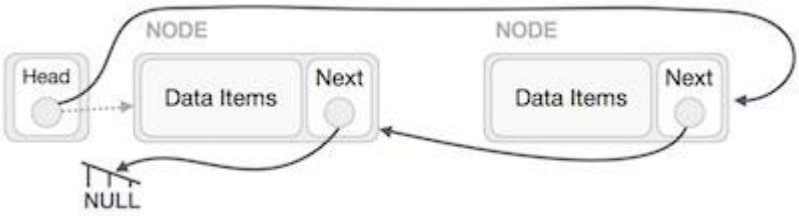
First, we traverse to the end of the list. It should be pointing to NULL. Now, we shall make it point to its previous node −



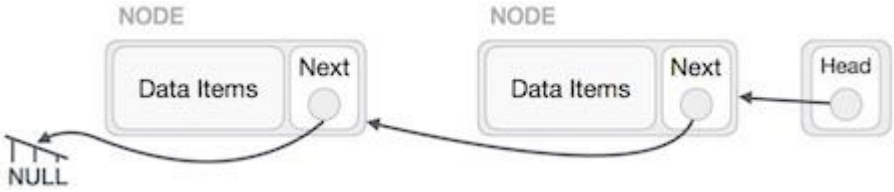
We have to make sure that the last node is not the last node. So we'll have some temp node, which looks like the head node pointing to the last node. Now, we shall make all left side nodes point to their previous nodes one by one.



Except the node (first node) pointed by the head node, all nodes should point to their predecessor, making them their new successor. The first node will point to NULL.



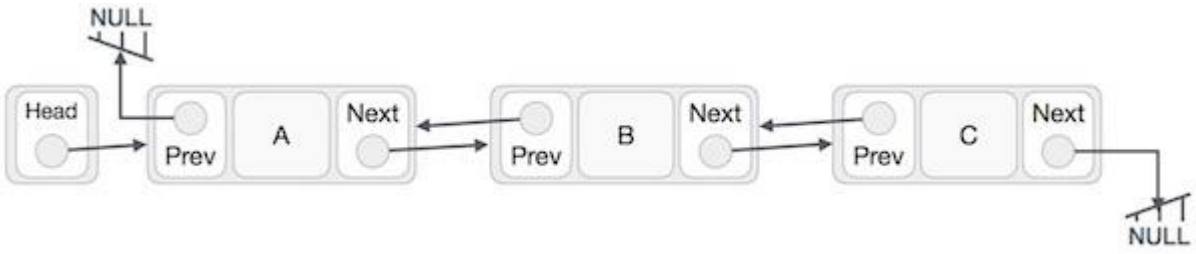
We'll make the head node point to the new first node by using the temp node.



Doubly Linked List is a variation of Linked list in which navigation is possible in both ways, either forward and backward easily as compared to Single Linked List. Following are the important terms to understand the concept of doubly linked list.

- **Link** − Each link of a linked list can store a data called an element.
- **Next** − Each link of a linked list contains a link to the next link called Next.
- **Prev** − Each link of a linked list contains a link to the previous link called Prev.
- **LinkedList** − A Linked List contains the connection link to the first link called First and to the last link called Last.

Doubly Linked List Representation



As per the above illustration, following are the important points to be considered.

- Doubly Linked List contains a link element called first and last.
- Each link carries a data field(s) and two link fields called next and prev.
- Each link is linked with its next link using its next link.
- Each link is linked with its previous link using its previous link.
- The last link carries a link as null to mark the end of the list.

Basic Operations

Following are the basic operations supported by a list.

- **Insertion** − Adds an element at the beginning of the list.
- **Deletion** − Deletes an element at the beginning of the list.
- **Insert Last** − Adds an element at the end of the list.
- **Delete Last** − Deletes an element from the end of the list.
- **Insert After** − Adds an element after an item of the list.
- **Delete** − Deletes an element from the list using the key.
- **Display forward** − Displays the complete list in a forward manner.
- **Display backward** − Displays the complete list in a backward manner.

Insertion Operation

Following code demonstrates the insertion operation at the beginning of a doubly linked list.

Example

```
//insert link at the first location
void insertFirst(int key, int data) {

   //create a link
   struct node *link = (struct node*) malloc(sizeof(struct
node));
   link->key = key;
   link->data = data;

   if(isEmpty()) {
      //make it the last link
      last = link;
   } else {
      //update first prev link
      head->prev = link;
   }

   //point it to old first link
   link->next = head;

   //point first to new first link
   head = link;
}
```

Deletion Operation

Following code demonstrates the deletion operation at the beginning of a doubly linked list.

Example

```
//delete first item
struct node* deleteFirst() {

   //save reference to first link
   struct node *tempLink = head;

   //if only one link
   if(head->next == NULL) {
      last = NULL;
   } else {
      head->next->prev = NULL;
   }

   head = head->next;

   //return the deleted link
   return tempLink;
}
```

Insertion at the End of an Operation

Following code demonstrates the insertion operation at the last position of a doubly linked list.

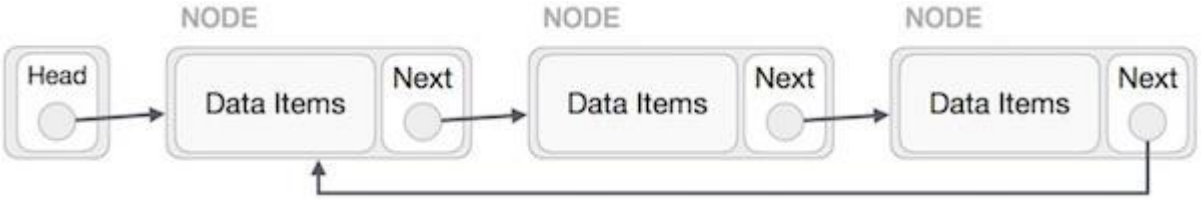
Example

```
//insert link at the last location
void insertLast(int key, int data) {

   //create a link
   struct node *link = (struct node*) malloc(sizeof(struct
node));
   link->key = key;
   link->data = data;

   if(isEmpty()) {
      //make it the last link
      last = link;
   } else {
      //make link a new last link
      last->next = link;

      //mark old last node as prev of new link
      link->prev = last;
   }

   //point last to new last node
   last = link;
}
```

Circular Linked List is a variation of Linked list in which the first element points to the last element and the last element points to the first element. Both Singly Linked List and Doubly Linked List can be made into a circular linked list.

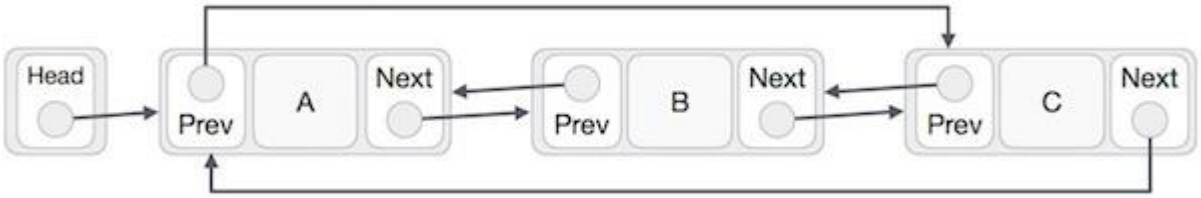
Singly Linked List as Circular

In singly linked list, the next pointer of the last node points to the first node.



Doubly Linked List as Circular

In doubly linked list, the next pointer of the last node points to the first node and the previous pointer of the first node points to the last node making the circular in both directions.



As per the above illustration, following are the important points to be considered.

- The last link's next points to the first link of the list in both cases of singly as well as doubly linked list.
- The first link's previous points to the last of the list in case of doubly linked list.

Basic Operations

Following are the important operations supported by a circular list.

- **insert** − Inserts an element at the start of the list.
- **delete** − Deletes an element from the start of the list.
- **display** − Displays the list.

Insertion Operation

Following code demonstrates the insertion operation in a circular linked list based on single linked list.

Example

```
insertFirst(data):
Begin
```

```
   create a new node
   node -> data := data
   if the list is empty, then
      head := node
      next of node = head
   else
      temp := head
      while next of temp is not head, do
      temp := next of temp
      done
      next of node := head
      next of temp := node
      head := node
   end if
End
```

Deletion Operation

Following code demonstrates the deletion operation in a circular linked list based on single linked list.

```
deleteFirst():
Begin
   if head is null, then
      it is Underflow and return
   else if next of head = head, then
      head := null
      deallocate head
   else
      ptr := head
      while next of ptr is not head, do
         ptr := next of ptr
      next of ptr = next of head
      deallocate head
      head := next of ptr
   end if
End
```

Display List Operation

Following code demonstrates the display list operation in a circular linked list.

```
display():
Begin
   if head is null, then
      Nothing to print and return
   else
      ptr := head
      while next of ptr is not head, do
         display data of ptr
         ptr := next of ptr
```

```
      display data of ptr
   end if
End
```

MCQs on Linked list with answers

1. 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.
View Answer / Hide Answer

ANSWER: B

2. 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
View Answer / Hide Answer

ANSWER: A

3. 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
View Answer / Hide Answer

ANSWER: A

4. Consider an implementation of unsorted singly linked list. Suppose it has its representation with a head and tail pointer. 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
View Answer / Hide Answer

ANSWER: C

5. 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
View Answer / Hide Answer

6. Consider an implementation of unsorted doubly linked list. Suppose it has its representation with a head pointer and tail pointer. 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 end node of the linked list

a) I and II
b) I and III
c) I,II and III
d) I,II,III and IV
View Answer / Hide Answer

ANSWER: D

7. Consider an implementation of unsorted doubly 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 end node of the linked list

a) I and II
b) I and III
c) I,II and III
d) I,II,III and IV
View Answer / Hide Answer

ANSWER: B

8. Consider an implementation of unsorted circular 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 end node of the linked list

a) I and II
b) I and III
c) I, II, III and IV
d) None
View Answer / Hide Answer

ANSWER: D

9. Consider an implementation of unsorted circular doubly 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 end node of the linked list

a) I and II
b) I and III
c) I, II and III
d) I,II,III and IV
View Answer / Hide Answer

ANSWER: D

10. In linked list each node contain minimum of two fields. One field is data field to store the data second field is?

a) Pointer to character
b) Pointer to integer
c) Pointer to node
d) Node
View Answer / Hide Answer

ANSWER: C

11. 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) θ (n)
d) θ (1)
View Answer / Hide Answer

ANSWER: C

12. What would be the asymptotic time complexity to add an element in the linked list?

a) O(1)
b) O(n)
c) $O(n^2)$
d) None
View Answer / Hide Answer

ANSWER: B

13. 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) None
View Answer / Hide Answer

ANSWER: B

14. 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) None
View Answer / Hide Answer

ANSWER: A

15. The concatenation of two list can performed in O(1) time. Which of the following variation of linked list can be used?

a) Singly linked list
b) Doubly linked list
c) Circular doubly linked list
d) Array implementation of list
View Answer / Hide Answer

ANSWER: C

16. Consider the following definition in c programming language

```
struct node
{
```

```
int data;
struct node * next;
}
typedef struct node NODE;
NODE *ptr;
```

Which of the following c code is used to create new node?

a) ptr=(NODE*)malloc(sizeof(NODE));
b) ptr=(NODE*)malloc(NODE);
c) ptr=(NODE*)malloc(sizeof(NODE*));
d) ptr=(NODE)malloc(sizeof(NODE));
View Answer / Hide Answer

ANSWER: A

17. 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
View Answer / Hide Answer

ANSWER: C

18. In doubly linked lists, traversal can be performed?

a) Only in forward direction

b) Only in reverse direction

c) In both directions

d) None

View Answer / Hide Answer

ANSWER: C

19. 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

View Answer / Hide Answer

20. A variation of linked list is circular linked list, in which the last node in the list points to first node of the list. One problem with this type of list is?

a) It waste memory space since the pointer head already points to the first node and thus the list node does not need to point to the first node.

b) It is not possible to add a node at the end of the list.

c) It is difficult to traverse the list as the pointer of the last node is now not NULL

d) All of above

View Answer / Hide Answer

ANSWER: C

21. A variant of the linked list in which none of the node contains NULL pointer is?

a) Singly linked list
b) Doubly linked list
c) Circular linked list
d) None
View Answer / Hide Answer

ANSWER: C

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

a) One pointer
b) Two pointer
c) Three pointer
d) None
View Answer / Hide Answer

ANSWER: B

23. Which of the following statements about linked list data structure is/are TRUE?

a) Addition and deletion of an item to/ from the linked list require modification of the existing pointers
b) The linked list pointers do not provide an efficient way to search an item in the linked list
c) Linked list pointers always maintain the list in ascending order
d) The linked list data structure provides an efficient way to find kth element in the list
View Answer / Hide Answer

ANSWER: B

24. Linked lists are not suitable to for the implementation of?

a) Insertion sort
b) Radix sort
c) Polynomial manipulation
d) Binary search
View Answer / Hide Answer

ANSWER: D

25. In worst case, the number of comparison need to search a singly linked list of length n for a given element is

a) log n
b) n/2
c) $\log_2 n-1$
d) n
View Answer / Hide Answer

ANSWER: D

26. consider the function f defined here:

struct item

```
{
int data;
struct item * next;
};
int f (struct item *p)
{
return((p==NULL) ||((p->next==NULL)||(p->data<=p->next->data) && (p->next)));
}
```

For a given linked list p, the function f returns 1 if and only if

a) the list is empty or has exactly one element
b) the element in the list are sorted in non-decreasing order of data value
c) the element in the list are sorted in non-increasing order of data value
d) not all element in the list have the same data value
View Answer / Hide Answer

ANSWER: B

27. The following C function takes a singly linked list as input argument. It modifies the list by moving the last element to the front of the list and returns the modified list. Some part of the code left blank.

```
typedef struct node
{
int value;
struct node* next;
}Node;
Node* move_to_front(Node* head)
{
Node* p, *q;
if((head==NULL) || (head->next==NULL))
return head;
```

```
q=NULL;
p=head;
while(p->next != NULL)
{
q=p;
p=p->next;
}
return head;
}
```

Choose the correct alternative to replace the blank line

a) q=NULL; p->next=head; head =p ;
b) q->next=NULL; head =p; p->next = head;
c) head=p; p->next=q; q->next=NULL;
d) q->next=NULL; p->next=head; head=p;
View Answer / Hide Answer

ANSWER: D

28. The following C Function takes a singly- linked list of integers as a parameter and rearranges
the elements of the lists. The function is called with the list containing the integers 1,2,3,4,5,6,7 in the given order. What will be the contents of the list after the function completes execution?

```
struct node{
int value;
struct node* next;
};
void rearrange (struct node* list)
{
struct node *p,q;
```

```
int temp;
if (! List || ! list->next) return;
p->list; q=list->next;
while(q)
{
temp=p->value; p->value=q->value;
q->value=temp;p=q->next;
q=p?p->next:0;
}
}
```

a) 1, 2, 3, 4, 5, 6, 7

b) 2, 1, 4, 3, 6, 5, 7

c) 1, 3, 2, 5, 4, 7, 6

d) 2, 3, 4, 5, 6, 7, 1

View Answer / Hide Answer

ANSWER: B