1. Two main measures for the efficiency of an algorithm are

a. Processor and memory  
b. Complexity and capacity  
**c. Time and space**d. Data and space

2.The Worst case occur in linear search algorithm when

a. Item is somewhere in the middle of the array  
**b. Item is not in the array at all**c. Item is the last element in the array  
d. Item is the last element in the array or is not there at all

3. The Average case occur in linear search algorithm

**a. When Item is somewhere in the middle of the array**   
b. When Item is not in the array at all  
c. When Item is the last element in the array  
d. When Item is the last element in the array or is not there at all

4. The complexity of linear search algorithm is

**a. O(n)**b. O(log n)  
c. O(n2)  
d. O(n log n)

5. The complexity of Bubble sort algorithm is

**a. O(n)**  
b. O(log n)  
c. O(n2)  
d. O(n log n)

6.The complexity of Binary search algorithm is

a. O(n)  
**b. O(log n )**  
c. O(n2)  
d. O(n log n)

7. Which of the following data structure is linear data structure?

a. Trees  
b. Graphs  
**c. Arrays**  
d. None of above

8. What are Abstract Data Type?  
a) They are a set of values and operations for manipulating those values  
b) They are a scheme for storing values in computer memory  
c) Arrays, stacks, queues,lists, and trees are all examples of abstract data types  
d) a, b  
**e) a, c**

Answer:E

9. Data structures generally employ which of the following implementation strategies?  
a) Contiguous Implementation  
b) Linked Implementation  
c) All of the mentioned

Answer:C

10. What is Linked Implementation?  
a) Values are stored in adjacent memory cells  
b) Values are not necessarily stored in adjacent memory cells and are accessed using pointers or references  
c) Values are not stored in adjacent memory cells

Answer:B

11. Which one of the below is not divide and conquer approach?

[**A** - Insertion Sort](javascript:void(0);)

[**B** - Merge Sort](javascript:void(0);)

[**C** - Shell Sort](javascript:void(0);)

[**D** - Heap Sort](javascript:void(0);)

Answer:B

12. If the array is already sorted, which of these algorithms will exhibit the best performance

[**A** - Merge Sort](javascript:void(0);)

**B** - Insertion Sort

[**C** - Quick Sort](javascript:void(0);)

[**D** - Heap Sort](javascript:void(0);)

Answer:B

13. How many swaps are required to sort the given array using bubble sort - { 2, 5, 1, 3, 4}

**A** - 4

[**B** - 5](javascript:void(0);)

[**C** - 6](javascript:void(0);)

[**D** - 7](javascript:void(0);)

Answer:A

14. Which of the following is not possible with an array in C programming langauge −

[**A** - Declaration](javascript:void(0);)

[**B** - Definition](javascript:void(0);)

**C** - Dynamic Allocation

[**D** - Array of strings](javascript:void(0);)

Answer:C

15. Which of the following sorting methods would be most suitable for sorting a list which is almost sorted  
  
a. Bubble Sort   
b. Insertion Sort  
c. Selection Sort   
d. Quick Sort

Answer:A

16. Which of the following is not a limitation of binary search algorithm?  
  
a. must use a sorted array  
b. requirement of sorted array is expensive when a lot of insertion and deletions are needed  
c. there must be a mechanism to access middle element directly  
d. binary search algorithm is not efficient when the data elements are more than 1000.

Answer:D

## 17. An empty list is the one which has no

a.nodes  
b.data  
c.both a and b  
d.address  
  
Answer:c

18. **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.

**ANSWER: B**

**19. 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

**ANSWER: A**

**20. 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

**ANSWER: C**

**21. 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

**ANSWER: C**

**22. What would be the asymptotic time complexity to add an element in the linked list?**  
  
a) O(1)  
b) O(n)  
c) O(n2)  
d) None

**ANSWER: B**

**23.** 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));

**ANSWER: A**

**24.** Which of the following points is/are true about Linked List data structure when it is compared with array

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

1. Arrays have better cache locality that can make them better in terms of performance
2. It is easy to insert and delete elements in Linked List
3. Random access is not allowed in a typical implementation of Linked Lists
4. All of the above

**ANSWER: D**

**25.** Consider the following function that takes reference to head of a Doubly Linked List as parameter. Assume that a node of doubly linked list has previous pointer as *prev*and next pointer as *next*.

|  |
| --- |
| void fun(struct node \*\*head\_ref)  {      struct node \*temp = NULL;      struct node \*current = \*head\_ref;        while (current !=  NULL)      {          temp = current->prev;          current->prev = current->next;          current->next = temp;          current = current->prev;      }        if(temp != NULL )          \*head\_ref = temp->prev;  } |

Assume that reference of head of following doubly linked list is passed to above function 1 <--> 2 <--> 3 <--> 4 <--> 5 <-->6. What should be the modified linked list after the function call?

a .2 <--> 1 <--> 4 <--> 3 <--> 6 <-->5

b. 5 <--> 4 <--> 3 <--> 2 <--> 1 <-->6

c. 6 <--> 5 <--> 4 <--> 3 <--> 2 <--> 1

d. 6 <--> 5 <--> 4 <--> 3 <--> 1 <--> 2

**ANSWER: C**

26. The following function reverse() is supposed to reverse a singly linked list. There is one line missing at the end of the function.

|  |
| --- |
| /\* Link list node \*/  struct node  {      int data;      struct node\* next;  };    /\* head\_ref is a double pointer which points to head (or start) pointer    of linked list \*/  static void reverse(struct node\*\* head\_ref)  {      struct node\* prev   = NULL;      struct node\* current = \*head\_ref;      struct node\* next;      while (current != NULL)      {          next  = current->next;          current->next = prev;          prev = current;          current = next;      } |

What should be added in place of "/\*ADD A STATEMENT HERE\*/", so that the function correctly reverses a linked list.

1. \*head\_ref = prev;
2. \*head\_ref = current;
3. \*head\_ref = next
4. \*head\_ref = NULL

Answer:A

27. What is the output of following function for start pointing to first node of following linked list? 1->2->3->4->5->6

|  |
| --- |
| void fun(struct node\* start)  {    if(start == NULL)      return;    printf("%d  ", start->data);      if(start->next != NULL )      fun(start->next->next);    printf("%d  ", start->data);  } |

1. 1 4 6 6 4 1
2. 1 3 5 1 3 5
3. 1 2 3 5
4. 1 3 5 5 3 1

Answer:D

28. n 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

Answer:D

29. To find out maximum element in a list of n numbers, one needs at least

**A** n comparisons

**B** n-1 comparisons

**C** n(n-1) com

Answer:B

30. In a doubly linked list traversing comes to a halt at:

**A** null

**B** front

**C** rear

Answer:A

**31.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;**

32. **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

33. 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);

}

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

1. Prints all nodes of linked lists
2. **Prints all nodes of linked list in reverse order**
3. Print s alternate nodes of list
4. Prints alternate nodes in reverse order

34. **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

35. What is the output of following function for start pointing to first node of following linked list? 1->2->3->4->5->6

|  |
| --- |
| void fun(struct node\* start)  {    if(start == NULL)      return;    printf("%d  ", start->data);      if(start->next != NULL )      fun(start->next->next);    printf("%d  ", start->data);  } |

1. 1 4 6 6 4 1
2. 1 3 5 1 3 5
3. 1 2 3 5
4. **1 3 5 5 3 1**