Q1. What is Data Structure?
Ans: A data structure is an arrangement of data in a computer's memory or even disk storage. It has a different way of storing and organizing data in a computer so that it can used efficiently. Different kind of Data structure is used in different application. For example, B-tree is widely used in implementation of databases.

Q2. Name different type of data structures?
Ans: There are basically two types of data structures

- Linear data Structure
- Non Linear data Structure

Q3. What is Linear Data Structure ?
Ans: Data structure where data elements are arranged sequentially or linearly where the elements are attached to its previous and next adjacent in what is called a **linear data structure**. In linear data structure, single level is involved. Therefore, we can traverse all the elements in single run only. Linear data structures are easy to implement because computer memory is arranged in a linear way. Its examples are array, stack, queue, linked list, etc.

Q.4 What is Non-Linear Data Structure?
Ans: Non Linear Data Structures are constructed by attaching a data element to several other data elements in such a way that it reflects a specific relationship among them For Example: – N-dimensional Array, Trees and Graphs

Arrays
Q.1 What is an Array?
Ans: An Array can be defined as a data structure having fixed size sequenced collection of elements of same data type and which can be referenced through same variable name. An Array is a name given to a group of similar quantities.
Declaring an array
Type variable_name [length of array];
For example:-
double height[10];
float width[20];
int min[9];
int myArray[5] = {1, 2, 3, 4, 5}; //declare and initialize the array in one statement
In C language array starts at position 0. C language treats array name as the pointer to the first element and an item in the array can be accessed through its index.

Q.2 How to read and write data in an array?
Ans:
```
#include<stdio.h>
void main ()
{
 int array[10], n;

 // Reading an array
 printf("Enter size of an array");
 scanf ("%d", &n);
 printf("Enter array elements");
 for (i=0;i<n;i++)
 scanf ("%d", &array[i]);

 // writing an array
 printf("array elements are");
 for (i=0;i<n;i++)
 printf ("%d", array[i]);
}
```

Q.3 How to insert data in an array at a particular position?
Ans: To insert a value into an array, you need to shift every value after the location you want to insert. The logic is very simple: start from the end and copy the previous item to the current element until you hit the location you want to insert.

Q.4 Write code to insert data in an array at a particular position.
Ans:
```
#include< stdio.h >
 void main ()
 {
  int array [10], n elem, pos;
 // Reading an array
  printf ("Enter size of an array");
  scanf ("%d", &n);
  printf ("Enter array elements");
  for (i=0;i<n;i++)
  scanf ("%d", &array[i]);
  //Inserting an element at specified position
  printf ("Enter element to be inserted");
  scanf ("%d", &elem);
  printf("Enter position where to insert the element");
  scanf ("%d", &pos);
  pos--;
 for (i=pos;i--)
  a[i+1]=a[i];
  a[pos]=elem;
  n++;
 // writing an array
  printf ("array elements are");
  for (i=0;i<n;i++)
  printf ("%d", array[i]);
 }
```

Q.5 How to delete an element from specified position from an array?
Ans: To delete a value from an array, you need to shift every value after the location you want to delete. The logic is very simple: start from the position you want to delete and copy the previous item to the current element until end of the array.

Q.6 Write code to delete data from an array from a particular position?
Ans:
```
#include< stdio.h >
 void main ()
 {
  int array[10], n elem ,pos;
// Reading an array
  printf("Enter size of an array");
  scanf ("%d", &n);
  printf("Enter array elements");
  for (i=0;i<n;i++)
  scanf ("%d", &array[i]);
 //deleting an element at specified position
  printf("Enter position from where to delete the element");
  scanf ("%d", &pos);
  pos--;
 for (i=pos ; i<n ;i++)
  a[i]= a[i+1];
  n--;
// writing an array
  printf("array elements are");
  for (i=0;i<n;i++)
  printf ("%d", array[i]);
 }
```

Q.7 How to merge two sorted arrays?
Ans: Assume, that both arrays are sorted in ascending order and we want resulting array to maintain the same order. Algorithm to merge two arrays A[0..m-1] and B[0..n-1] into an array C[0..m+n-1] is as following:

- Introduce read-indices i , j to traverse arrays A and B, accordingly. Introduce write-index k to store position of the first free cell in the resulting array. By default i = j = k = 0.
- At each step: if both indices are in range (i < m and j < n), choose minimum of (A[i], B[j]) and write it to C[k]. Otherwise go to step 4.
- Increase k and index of the array, algorithm located minimal value at, by one. Repeat step 2.
- Copy the rest values from the array, which index is still in range, to the resulting array.

Q.9 What is two-dimensional array?
Ans: Two-dimensional arrays, the most common multidimensional arrays, are used to store information that we normally represent in table form. Two-dimensional arrays, like one-dimensional arrays, are homogeneous. This means that all of the data in a two-dimensional array is of the same type. The two-dimensional array may also be visualized as a one-dimensional array of arrays. Examples of applications involving two-dimensional arrays include:

- a seating plan for a room (organized by rows and columns),
- a monthly budget (organized by category and month), and
- a grade book where rows might correspond to individual students and columns to student scores.

Declaration of Two-Dimensional Arrays ***int a[3][2];***
We can declare and initialize an array A as follows:
```
//declaration
int A[3][4] = {{8, 2, 6, 5}, //row 0
{6, 3, 1 ,0}, //row 1
{8, 7, 9, 6}}; //row 2
```

Searching

Q .1 How to do Linear Search in an Array?

Ans: Linear search or Sequential is the simplest search algorithm for finding a particular value in an array that consists of checking every one of its elements, one at a time and in sequence, until the desired one is found.

Q .2 Write code to do Linear Search in an Array?

Ans:

```
/* Program for linear search an element */

#include <stdio.h>
#include <conio.h>

void main()
{
    int array[10];
    int i, N, keynum, found=0;
    clrscr();

    printf("Enter the value of N\n");
    scanf("%d",&N);

    printf("Enter the elements one by one\n");

for(i=0; i<N ; i++)
                scanf("%d",&array[i]);
            printf("Input array is\n");
            for(i=0; i<N ; i++)
                printf("%d\n",array[i]);

            printf("Enter the element to be searched\n");
            scanf("%d", &keynum);
        /* Linear search begins */
        i=0;
        While(i < n) && (found==0)
        {
            if( keynum == array[i] )
                found = 1;
            i++;
        }
        if ( found == 1)
            printf("SUCCESSFUL SEARCH\n");
        else
            printf("Search is FAILED\n");
        } /* End of main */
```

Q.3 How to do Binary Search in an Array?

Ans: A Binary Search or Half-interval search algorithm finds the position of a specified value (the input "key") within a sorted array. In each step, the algorithm compares the input key value with the key value of the middle element of the array. If the keys match, then a matching element has been found so its index, or position, is returned. Otherwise, if the sought key is less than the middle element's key, then the algorithm repeats its action on the sub-array to the left of the middle element or, if the input key is greater, on the sub-array to the right. If the remaining array to be searched is reduced to zero, then the key cannot be found in the array and a special "Not found" indication is returned.

Q.4 Write 'C' code to do Binary Search in an Array?

Ans:

```
#include <stdio.h>
#include <conio.h>

void main()
{
    int array[10];
    int i, n, elem, found=0;
    clrscr();

    printf("Enter the value of N\n");
    scanf("%d",&n);

    printf("Enter the elements one by one\n");
    for(i=0; i<n ; i++)
        scanf("%d",&array[i]);
    printf("Input array is\n");
    for(i=0; i<n; i++)
        printf("%d\n",array[i]);

    printf("Enter the element to be searched\n");
    scanf("%d", &elem);
/* Binary search begins */
l=0;
        u=n-1;
        While((l<u) && (found==0))
        {
            mid=(l+u)/2;
            if(a[mid]==elem)
            found=1;
        else
            if(a[mid]>elem)
                u=mid-1;
            else
                l=mid+1;
        }
        mid=mid+1;
    if ( found == 1)
            printf("SUCCESSFUL SEARCH\n");
        else
            printf("Search is FAILED\n");
 } /* End of main */
```

Stack

Q1. What is Stack ?

Ans: Stack is a linear data structure which follows a particular order in which the operations are performed. The order may be LIFO(Last In First Out) or FILO(First In Last Out).

Q.2 Write a Program for Stack implementation through Array?

Ans:

```
#include <stdio.h>
#include<ctype.h>
# define MAXSIZE 200

int stack[MAXSIZE];
int top; //index pointing to the top of stack
void main()
{
void push(int);
int pop();
int will=1,i,num;
while(will ==1)
{
    printf("MAIN MENU:
    1.Add element to stack
    2.Delete element from the stack
");
Scanf ("%d", &will);

switch(will)
{
case 1: printf("Enter the data... ");
            scanf("%d", &num);
            push(num);
            break;
        case 2: num=pop();
            printf("Value returned from pop function is %d ",num);
            break;
        default: printf ("Invalid Choice . ");
        }
        printf("Do you want to do more operations on Stack ( 1 for yes,
any other key to exit) ");
        scanf("%d" , &will);
        } //end of outer while
        }       //end of main
        void push(int elem)
        {
            if(isfull())
                {
                    printf("STACK FULL");
                    return;
                }
            top++;
            stack[top]=elem;
            }
        }
        int pop()
        {
```

```
    int a;
    if(isEmpty())
    {
        printf("STACK EMPTY");
        return 0;
    }
    a=stack[top];
top--;
}
return(a);
}

int isFull()
{
    if(top> MAXSIZE)
        return 0;
    else
        return 1;
}
int isEmpty()
{
if(top<=0)
        return 0;
else
        return 1;
}
```

Q.3 What is the main difference between ARRAY and STACK?
Ans: Stack follows LIFO. Thus the item that is first entered would be the last removed. In array the items can be entered or removed in any order. Basically each member access is done using index and no strict order is to be followed here to remove a particular element .Array may be multi-dimensional or one dimensional but stack should be one-dimensional. Size of array is fixed, while stack can be grow or shrink. We can say stack is dynamic data structure.

Q.4 What are various Applications of Stack?
Ans: Some of the applications of stack are:

1. Evaluating arithmetic expression
2. Balancing the symbols
3. Towers of Hanoi
4. Function Calls
5. 8 Queen Problem.

Queue
Q1. What is Queue?
Ans: A Queue is a linear structure which follows a particular order in which the operations are performed. The order is First In First Out (FIFO). A good example of a queue is any queue of consumers for a resource where the consumer that came first is served first. The difference between stacks and queues is in removing. In a stack we remove the item the most recently added; in a queue, we remove the item the least recently added.
Q.2 Write a code in 'C' for Linear Queue through Array?
Ans:

```
#include <stdio.h>
#include<ctype.h>
# define MAXSIZE 200

int queue[MAXSIZE];
int front=-1, rear=-1; //index pointing to the top of stack
void main()
{
void enqueue(int);
int dequeue();
int will=1,i,num,ch;
while(will ==1)
{
    printf("MAIN MENU:
    1. Enqueue
    2.Dequeue
    3. Check Queue Full
    4. Check Queue Empty ");

    Scanf ("%d", &ch);

switch(ch)
        {
        case 1: printf("Enter the data to add... ");
            scanf("%d", &num);
            enqueue(num);
            break;
        case 2: num=dequeue();
            if(num==-1)
                printf("Queue is empty");\
            else
                printf("Value returned from pop function is %d ",num);
            break;
        case 3 : isFull();
            break;
        case 4 : isEmpty();
            break;
        default: printf ("Invalid Choice . ");
    }
    printf("Do you want to do more operations on Stack ( 1 for yes,
any other key to exit) ");
    scanf("%d" , &will);
    } //end of outer while
}       //end of main
void enqueue (int elem)
{
if(isfull())
    {
printf("QUEUE FULL");
        return;
    }
else{
    if(front==-1)
        front=rear=0;
    else
        rear++;
    queue[rear]=elem;
}
}
int dequeue()
{
int elem;
if(isEmpty())
{
    return -1;
}
else
{
    elem=queue[front];
    if (front==rear)
        front=rear=-1;
    else
        front++;
}
return(elem);
}
int isFull()
{
if(rear==MAXSIZE-1)
    return 0;
else
    return 1;
}
int isEmpty()
{
if((front==-1) && (front==rear))
    return 0;
else
    return 1;
}
```

Q.3 What is a Circular Queue?
Ans. A circular queue is one in which the insertion of new element is done at the very first location of the queue if the last location of the queue is full. Suppose if we have a Queue of n elements then after adding the element at the last index i.e. (n-1) th , as queue is starting with 0 index, the next element will be inserted at the very first location of the queue which was not possible in the simple linear queue. That's why linear queue leads to wastage of the memory, and this flaw of linear queue is overcome by circular queue. So, in all we can say that the circular queue is a queue in which first element come right after the last element that means a circular queue has a starting point but no end point.

Q.5 What is Priority Queue?
Ans: A Priority Queue is a collection of elements such that each element has been assigned a priority and such that the order in which each elements are deleted and processed comes from the following rules:

1. An element of higher priority is processes before any element of lower priority.
2. Two elements with the same priority are processed according to the order in which they were added to the queue.

A priority queue must at least support the following operations:

1. **insert_with_priority:** add an element to the queue with an associated priority
2. **pull_highest_priority_element:** remove the element from the queue that has the highest priority , and return it.

Algorithm to add an item in priority queue
This algorithm adds an item with priority number M to a priority Queue maintained by a 2-dimensional array Queue.

1. Insert Item as the rear element in row M of Queue.
2. Exit.

Algorithm to delete an item from a priority queue
This algorithm deleted and processes the first element in a priority queue maintained by a 2-dimensional array Queue

1. [Find the first nonempty queue]
2. Find the smallest K such that FRONT[K]!=NULL.
3. Delete and process the front element in row K of Queue.
4. Exit.

Sorting

Q.1 What is Sorting?

Ans: Sorting refers to ordering data in an increasing or decreasing fashion according to some linear relationship among the data items.

Q.2 Write a 'C' code for selection sort?

Ans:

```
#include<stdio.h>
        main ()
        {
            int array[100], size, i, j, min, swap;

            printf("Enter number of elements\n");
            scanf("%d", &size);

            printf("Enter %d integers\n", size);
            for ( i = 0 ; i < size ; i++ )
                scanf("%d", &array[i]);
            for ( i = 0 ; i < ( size - 1 ) ; i++ )
            {
                min = i;
                for ( j = i + 1 ; j < size ; j++ )
                {
                    if ( array[min] > array[j] )
                    min = j;
                }
                if ( min != i )
                {
swap = array[i];
                    array[i] = array[min];
                    array[min] = swap;
            }
}
printf("Sorted list in ascending order:\n");
for ( i = 0 ; i < size ; i++ )
            printf("%d\n", array[i]);
return 0;
}
```

Q.3 What is bubble sort?

Ans: Bubble sort , often incorrectly referred to as sinking sort , is a simple sorting algorithm that works by repeatedly stepping through the list to be sorted, comparing each pair of adjacent items and swapping them if they are in the wrong order. The pass through the list is repeated until no swaps are needed, which indicates that the list is sorted. The algorithm gets its name from the way smaller elements "bubble" to the top of the list. Because it only uses comparisons to operate on elements, it is a comparison sort.

For Example: Let us take the array of numbers "5 1 4 2 8", and sort the array from lowest number to greatest number using bubble sort algorithm. In each step, elements written in **bold** are being compared. Three passes will be required.

First Pass:

(**5 1** 4 2 8) → (**1 5** 4 2 8), Here, algorithm compares the first two elements, and swaps them.

(1 **5 4** 2 8) → (1 **4 5** 2 8), Swap since 5 > 4

(1 4 **5 2** 8) → (1 4 **2 5** 8), Swap since 5 > 2

(1 4 2 **5 8**) → (1 4 2 **5 8**), Now, since these elements are already in order (8 > 5), algorithm does not swap them.

Second Pass:

(**1 4** 2 5 8) → (**1 4** 2 5 8)

(1 **4 2** 5 8) → (1 **2 4** 5 8), Swap since 4 > 2

(1 2 **4 5** 8) → (1 2 **4 5** 8)

(1 2 4 **5 8**) → (1 2 4 **5 8**) Now, the array is already sorted, but our algorithm does not know if it is completed. The algorithm needs one whole pass without any swap to know it is sorted.

Third Pass:

(**1 2** 4 5 8) → (**1 2** 4 5 8)

(1 **2 4** 5 8) → (1 **2 4** 5 8)

(1 2 **4 5** 8) → (1 2 **4 5** 8)

(1 2 4 **5 8**) → (1 2 4 **5 8**)

Q .4 Write a 'C' code for bubble sort?

Ans:

```
// Program of sorting using bubble sort method
#include <stdio.h>
#define MAX 20
main()
{
    int arr[MAX], i, j, k, temp, size, xchanges;
    printf("Enter the number of elements : ");
        scanf("%d",&n);

    printf("Enter array element");
    for (i = 0; i < n; i++)
        scanf("%d",&arr[i]);

    printf("Unsorted list is :\n");
    for (i = 0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");
/* Bubble sort*/
for (i = 1; i < n ; i++)
{
    for (j = 0; j <n-1-i; j++)
    {
        if (arr[j] > arr[j+1])
        {
            temp = arr[j];
            arr[j] = arr[j+1];
            arr[j+1] = temp;
        }/*End of if*/
    }/*End of inner for loop*/
}/*End of outer for loop*/
printf("Sorted list is :\n");
for (i = 0; i < n; i++)
    printf("%d ", arr[i]);
printf("\n");
}/*End of main()*/
```

Q.5 Write a 'C' code for insertion sort?

Ans:

```
#include<stdio.h>
int main()
{
    int i,j,size,temp,a[20];
    clrscr();
    printf("\nEnter size of the array: ");
    scanf("%d",&size);
    printf("\nEnter elements in to the array:");
    for(i=0;i<size;i++)
    scanf("%d",&a[i]);
    for(i=1;i<size;i++)
    {
        temp=a[i];
        j=i-1;
        while((j>=0)&&(temp<a[j]))
        {
            a[j+1]=a[j];
j=j-1;
        }
        a[j+1]=temp;
    }
    printf("\nAfter sorting the elements are: ");
    for(i=0;i<size;i++)
    printf(" %d",a[i]);
    return 0;
    }
```

Q.6 What is Merge sort?

Ans: The divide-and-conquer strategy is used in quicksort. Below the recursion step is described:

1. **Choose a pivot value.** We take the value of the middle element as pivot value, but it can be any value, which is in range of sorted values, even if it doesn't present in the array.
2. **Partition.** Rearrange elements in such a way, that all elements which are lesser than the pivot go to the left part of the array and all elements greater than the pivot, go to the right part of the array. Values equal to the pivot can stay in any part of the array. Notice, that array may be divided in non-equal parts.
3. **Sort both parts.** Apply quicksort algorithm recursively to the left and the right parts.

Partition algorithm in detail: There are two indices ***i*** and ***j*** and at the very beginning of the partition algorithm ***i*** points to the first element in the array and ***j*** points to the last one. Then algorithm moves ***i*** forward, until an element with value greater or equal to the pivot is found. Index ***j*** is moved backward, until an element with value lesser or equal to the pivot is found. If ***i ≤ j*** then they are swapped and i steps to the next position ***(i + 1)***, ***j*** steps to the previous one ***(j – 1)*** . Algorithm stops, when i becomes greater than ***j*** . After partition, all values before ***i-th*** element are less or equal than the pivot and all values after ***j-th*** element are greater or equal to the pivot.

Linked List

Q.1 Write an algorithm to search an element in a linear linked list?

Ans: Here START is a pointer variable which contains the address of first node. ITEM is the value to be searched.

1. Set PTR = START, LOC = 1 [Initialize PTR and LOC]
2. Repeat While (PTR != NULL)
3. If (ITEM == PTR->INFO) Then [Check if ITEM matches with INFO field]
4. Print: ITEM is present at location LOC
5. Return
6. Else
7. PTR = PTR->LINK [Move PTR to next node]
8. LOC = LOC + 1 [Increment LOC]
9. [End of If]
10. [End of While Loop]
11. Print: ITEM is not present in the list
12. Exit

Trees

Q.1 Explain binary search tree?

Ans: A tree is a finite set of one or more nodes such that:-There is a specially designated node called the root. The remaining nodes are partitioned into n>=0 disjoint sets T1, …, Tn, where each of these sets is a tree. We call T1, …, Tn the subtrees of the root.

Binary Trees

1. A binary tree is a finite set of nodes that is either empty or consists of a root and two disjoint binary trees called the left subtree and the right subtree.
2. Any tree can be transformed into binary tree.by left child-right.
3. The left subtree and the right subtree are distinguished.

Some Extra Questions

Q.1 What is data structure?

Ans: A data structure is a way of organizing data that considers not only the items stored, but also their relationship to each other. Advance knowledge about the relationship between data items allows designing of efficient algorithms for the manipulation of data.

Q.2 List out the areas in which data structures are applied extensively?

Ans: Compiler Design, Operating System, Database Management System, Statistical analysis package, Numerical Analysis, Graphics, Artificial Intelligence, Simulation

Q.3 What are the major data structures used in the following areas: RDBMS, Network data model & Hierarchical data model.

Ans:

1. RDBMS – Array (i.e. Array of structures)
2. Network data model – Graph
3. Hierarchical data model – Trees

Q.4 If you are using C language to implement the heterogeneous linked list, what pointer type will you use?

Ans: The heterogeneous linked list contains different data types in its nodes and we need a link, pointer to connect them. It is not possible to use ordinary pointers for this. So we go for void pointer. Void pointer is capable of storing pointer to any type as it is a generic pointer type.

Q.5 Minimum number of queues needed to implement the priority queue?

Ans: Two. One queue is used for actual storing of data and another for storing priorities.

Q.6 What is the data structures used to perform recursion?

Ans: Stack. Because of its LIFO (Last In First Out) property it remembers its "caller" so knows whom to return when the function has to return. Recursion makes use of system stack for storing the return addresses of the function calls. Every recursive function has its equivalent iterative (non-recursive) function. Even when such equivalent iterative procedures are written, explicit stack is to be used.

Q.7 What are the methods available in storing sequential files?

Ans: Straight merging, Natural merging, Polyphase sort, Distribution of Initial runs.

Q.8 List out few of the Application of tree data-structure?

Ans:

1. The manipulation of Arithmetic expression,
2. Symbol Table construction,
3. Syntax analysis.

Q.9 List out few of the applications that make use of Multilinked Structures?

Ans: Sparse matrix, Index generation.

Q.10 In tree construction which is the suitable efficient data structure?
(a) Array (b) Linked list (c) Stack (d) Queue (e) none

Ans: (b) Linked List

Q.11 What is the type of the algorithm used in solving the 8 Queens problem?

Ans: Backtracking

Q.12 In an AVL tree, at what condition the balancing is to be done?

Ans: If the "pivotal value" (or the "Height factor") is greater than 1 or less than –1.

Q.13 What is the bucket size, when the overlapping and collision occur at same time?

Ans: One. If there is only one entry possible in the bucket, when the collision occurs, there is no way to accommodate the colliding value. This results in the overlapping of values.

Q.14 There are 8, 15, 13, 14 nodes were there in 4 different trees. Which of them could have formed a full binary tree?

Ans: 15.

In general: There are 2n -1 nodes in a full binary tree.

By the method of elimination: Full binary trees contain odd number of nodes. So there cannot be full binary trees with 8 or 14 nodes, so rejected. With 13 nodes you can form a complete binary tree but not a full binary tree. So the correct answer is 15.

Note Full and Complete binary trees are different. All full binary trees are complete binary trees but not vice versa.

Q.15 Classify the Hashing Functions based on the various methods by which the key value is found.

Ans:

- Direct method
- Subtraction method
- Modulo-Division method
- Digit-Extraction method
- Mid-Square method
- Folding method,
- Pseudo-random method

Q.16 What are the types of Collision Resolution Techniques and the methods used in each of the type?

Ans:

1. Open addressing (closed hashing)
   The methods used include: Overflow block,
2. Closed addressing (open hashing)
   The methods used include: Linked list, Binary tree…

Q.17 In RDBMS, what is the efficient data structure used in the internal storage representation?

Ans: B+ tree. Because in B+ tree, all the data is stored only in leaf nodes, that makes searching easier. This corresponds to the records that shall be stored in leaf nodes.

Q.18 Of the following tree structure, which is, efficient considering space and time complexities?
(a) Incomplete Binary Tree
(b) Complete Binary Tree
(c) Full Binary Tree

Ans: (b) Complete Binary Tree.

By the method of elimination: Full binary tree loses its nature when operations of insertions and deletions are done. For incomplete binary trees, extra storage is required and overhead of NULL node checking takes place. So complete binary tree is the better one since the property of complete binary tree is maintained even after operations like additions and deletions are done on it.

Q.19 What is a spanning Tree?

Ans: A spanning tree is a tree associated with a network. All the nodes of the graph appear on the tree once. A minimum spanning tree is a spanning tree organized so that the total edge weight between nodes is minimized.

Q.20 Does the minimum spanning tree of a graph give the shortest distance between any 2 specified nodes?

Ans: No.Minimal spanning tree assures that the total weight of the tree is kept at its minimum. But it doesn't mean that the distance between any two nodes involved in the minimum-spanning tree is minimum.