Ans 1:-

```
for (i=0 to n)
{ if(arr[i] == value)
      // element found.
}
```

Ans 2:-

Iterative:-

```
Void insertion sort (int arr[], int n)
{
  for ( int i = 1; i<n ; i++ )
  {
    j = i-1 ;
    x = arr[i];
    while ( j > -1 && arr[j] > x )
    {
      arr[j+1] = arr[j];
      j--;
    }
    arr[j+1] = x;
  }
}
```

Recursive :-

```
void insertionsort (int arr[], int n)
{
    if (n<=1)
        return;
    insertion_sort (arr, n-1);
    int last = arr[n-1];
    int j = n-2;
    while (j>=0 && arr[j] > last)
    {
        arr[j+1] = arr[j];
        j--;
    }
    arr[j+1] = last;
}
```

Iterative sort is called 'Online sort' because it does not need to know anything about what value it will sort and info. is requested while algo is running.

Other Sorting Algorithms :-

1). Bubble sort
2). Quick sort
3). Merge sort
4). Selection sort
5). Heap sort.

Ans 3:-

Sorting Algo.	Best	Worst	Average
Selection sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Bubble sort	$O(n)$	$O(n^2)$	$O(n^2)$
Insertion sort	$O(n)$	$O(n^2)$	$O(n^2)$
Heap sort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
Quick sort	$O(n \log n)$	$O(n^2)$	$O(n \log n)$
Merge sort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$

Ans 4:-

Inplace sorting	Stable sorting	Online sorting
Bubble sort	Merge sort	Insertion sort
Selection sort	Bubble sort	
Insertion sort	Insertion sort	
Quick sort	Count sort	
Heap sort		

Ans :-

$$T(n) = T(n/2) + 1 \quad \text{— ①}$$

$$T(n/2) = T(n/4) + 1 \quad \text{— ②}$$

$$T(n/4) = T(n/8) + 1 \quad \text{— ③}$$

$$T(n) = T(n/2) + 1$$

$$= T(n/4) + 1 + 1$$

$$= T(n/8) + 1 + 1 + 1$$

$$\vdots$$

$$= T\left(\frac{n}{2^k}\right) + 1 \text{ (k times)}$$

Let $g^k = n$

$$k = \log n$$

$$T(n) = T\left(\frac{n}{n}\right) + \log n$$

$$T(n) = T(1) + \log n$$

$$T(n) = O(\log n)$$

Ans 7 :-

```
for (i=0; i<n; i++)
  {
    for (int j=0; j<n; j++)
      { if (a[i]+a[j]==k)
          printf("%d%d", i, j);
      }
  }
```

Ans 8:-

Quick sort is fastest general purpose sort. In most practical situations quicksort is the method of choice as stability is important and space is available, mergesort might be best.

Ans 9:- A pair (A[i], A[j]) is said to be inversions if

- A[i] > A[j]
- i < j

Total no. of inversions in given array are 31 using mergesort.

Ans 10:-

Worst case $O(n^2)$ → The worst case occurs when pivot element is an extreme (smallest / largest) element. This happens when input array is sorted or reverse sorted and either first or last element is selected as pivot.

Best Case $O(n \log n)$ → The Best case occurs when we will select pivot element as a mean element

Ans 11 :-

Merge sort :-

Best Case → $T(n) = 2T(n/2) + O(n)$
Worst Case → $T(n) = 2T(n/2) + O(n)$
} $O(n \log n)$

Quick Sort :-

Best Case → $T(n) = 2T(n/2) + O(n) \rightarrow O(n \log n)$
Worst Case → $T(n) = T(n-1) + O(n) \rightarrow O(n^2)$

In quick sort, array of elements is divided into two parts repeatedly until it is not possible to divide it further.

In mergesort, the elements are split into two subarray ($n/2$) again & again until only 1 element is left.

Ans 12:-

```
for (int i=0; i<n-1; i++)
  {
   int min = i;
   for (int j = i+1 ; j < n ; j++)
     { if (a[min] > a[j])
         min = j;
     }
   int key = a[min];
   while (min > i)
     {  a[min] = a[min - j];
        min --;
     }
   a[i] = key;
  }
```

Ans 13:-

A Better version of Bubble sort known as modified bubble sort, includes a flag that is a set of exchange is made after an entire pass over. If no exchange is made then it should be called array is already order because no two elements needs to be switched.