5.2.2 INSERTION SORT

Insertion sort is a simple sorting algorithm that works the way we sort playing cards in our hands. Insertion sorts works by taking element from the list one by one and inserting them in their current position into a new sorted list.

Algorithm:

1. Read the list of n elements to be sorted

2. Consider the first element as sorted and remaining element as unsorted.

3. Pick an element form unsorted part and insert it into the correct position in the sorted part.

4. Repeat this until all the elements are sorted.

5. Print the sorted list

Example:

Given n elements:

34 8 64 51 32 21

nd 34.

34 **8** 64 51 32 21

Select 8 and insert it into the sorted part

8 34 64 51 32 21-pass1

8 34 **64** 51 32 21-pass2 (64 is in proper position)

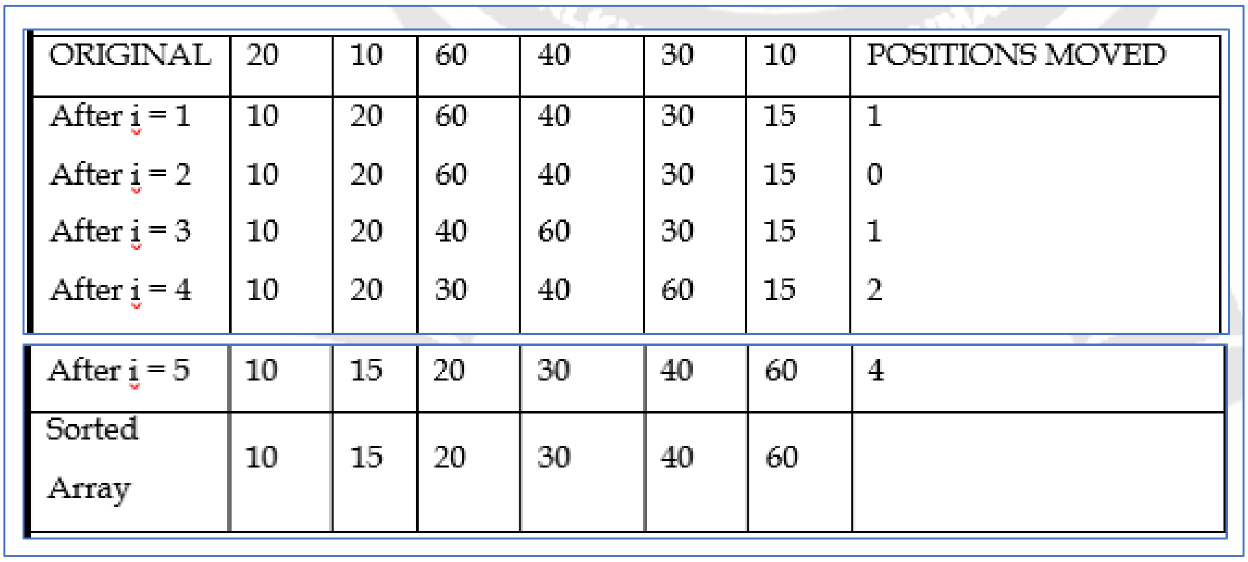
Select 51 and moves it to its appropriate location between 34 and 64.

8 **21** 32 34 51 64-pass5

Example:

Consider an unsorted array as follows,

20 10 60 40 30 15



ORIGINAL	20	10	60	40	30	10	POSITIONS MOVED
After i = 1	10	20	60	40	30	15	1
After i = 2	10	20	60	40	30	15	0
After i = 3	10	20	40	60	30	15	1
After i = 4	10	20	30	40	60	15	2
After i = 5	10	15	20	30	40	60	4
Sorted Array	10	15	20	30	40	60	

Program

```
void Insertion-sort(int a[], int n)

{

int i,j,temp; for(i=1; i<n; i++)
{

        temp = a[i]; j = i - 1;
        while((temp<a[j]) && (j>=0))

        {

                a[j+1] = a[j]; j = j - 1;
        }

        a[j+1] =temp;

}

cout<<"\n    Sorted
List  :”  ;  for(i=0;
i<n; i++)
        cout<< a[i];

}
```

Advantages of Insertion Sort

- Easy to **understand and implement**
- Works **well for small lists**
- **No extra memory** is needed (in-place sorting)
- **Stable sorting algorithm** (keeps order of equal elements)
- Efficient when the list is **already nearly sorted**

Disadvantages of Insertion Sort

- **Slow for large datasets**
- Time complexity is **$O(n^2)$** in average and worst cases
- Not suitable when **number of elements is large**
- More comparisons are needed

Applications of Insertion Sort

- Used to sort **small datasets**
- Used when data is **almost sorted**
- Used in **real-time systems** for small inputs
- Used as a **helper sort** in other algorithms
- Useful in **simple applications** where memory is limited