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.
- Insertion sort is a sorting algorithm that places an unsorted element at its suitable place in each iteration.
- This is an in-place comparison-based sorting algorithm. Here, a sub-list is maintained which is always sorted. For example, the lower part of an array is maintained to be sorted. An element which is to be 'insert'ed in this sorted sub-list, has to find its appropriate place and then it has to be inserted there. Hence the name, insertion sort.
- The array is searched sequentially and unsorted items are moved and inserted into the sorted sub-list (in the same array). This algorithm is not suitable for large data sets as its average and worst case complexity are of O(n2), where n is the number of items.

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	

Example-2

How Insertion Sort Works?

We take an unsorted array for our example.

Insertion sort compares the first two elements.

It finds that both 14 and 33 are already in ascending order. For now, 14 is in sorted sub-list.

Insertion sort moves ahead and compares 33 with 27.

And finds that 33 is not in the correct position.

It swaps 33 with 27. It also checks with all the elements of sorted sub-list. Here we see that the sorted sub-list has only one element 14, and 27 is greater than 14. Hence, the sorted sub-list remains sorted after swapping.

By now we have 14 and 27 in the sorted sub-list. Next, it compares 33 with 10.

These values are not in a sorted order.

So we swap them.

However, swapping makes 27 and 10 unsorted.

Hence, we swap them too.

Again we find 14 and 10 in an unsorted order.

We swap them again. By the end of third iteration, we have a sorted sub-list of 4 items.

Algorithm

Now we have a bigger picture of how this sorting technique works, so we can derive simple steps by which we can achieve insertion sort.

Step 1 − If it is the first element, it is already sorted. return 1;

Step 2 − Pick next element

Step 3 − Compare with all elements in the sorted sub-list

Step 4 − Shift all the elements in the sorted sub-list that is greater than the value to be sorted

Step 5 − Insert the value

Step 6 − Repeat until list is sorted

INSERTION SORT ROUTINE

```
Void insertionSort(int a[], int n)

{

int i, temp, j;

        for (i = 1; i < n; i++)

        {

        temp = a[i];

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

           {

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

           }

        a[j] = temp;

        }

}
```