Insertion sort

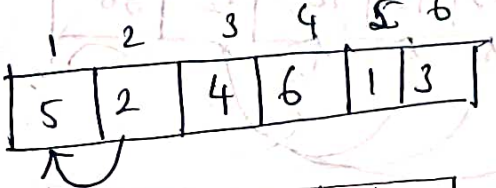
It is a simple sorting algorithm which sort the array by shifting one by one.

Characteristics of Insertion sort

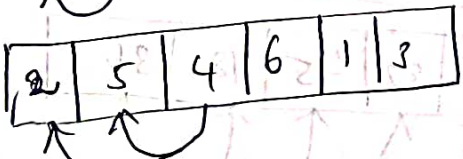
- It has one of the simplest Implementatio
- It is efficient for smaller data sets, but very inefficient for larger lists.
- Insertion sort is adaptive, that means it reduce its total number of steps if given a partially sorted list, hence it increase its efficiency.
- It is better than selection and bubble sort algorithms.
- It is stable, as it does not change the relative order of elements with equal keys.

Example

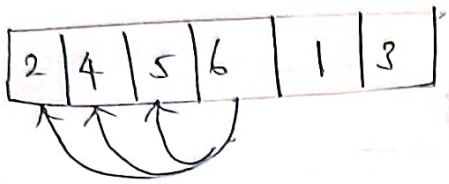
1.



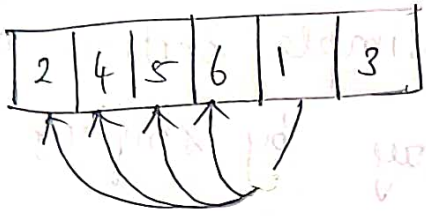
2.



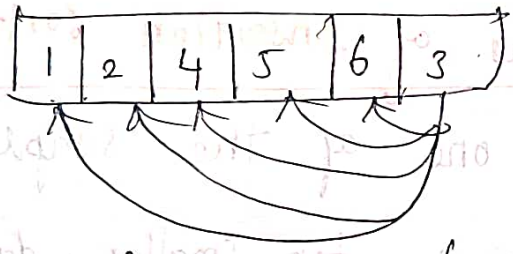
3.



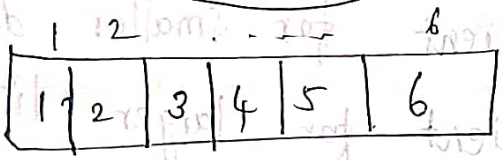
4.



5.

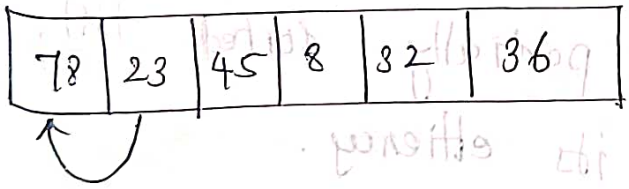


6.

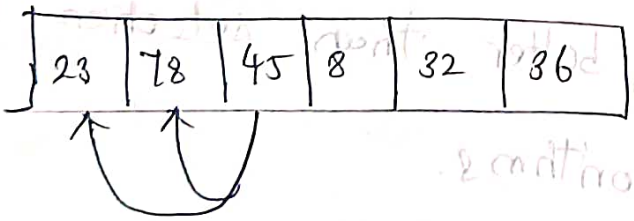


Example 2

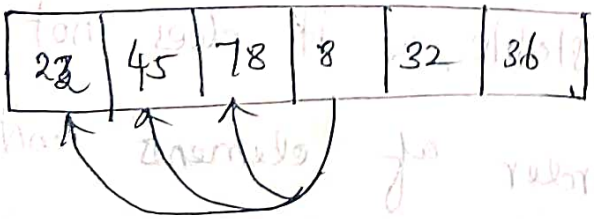
1.



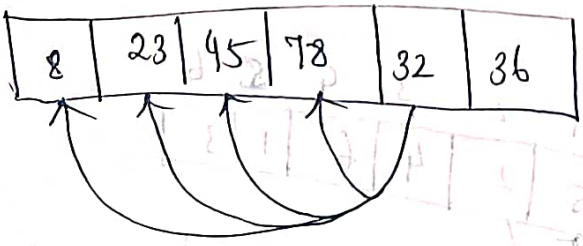
2.



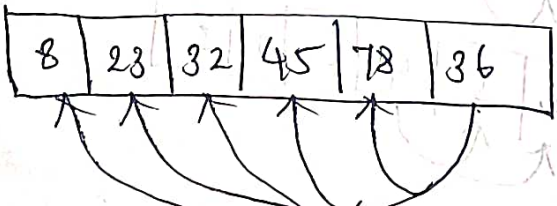
3.



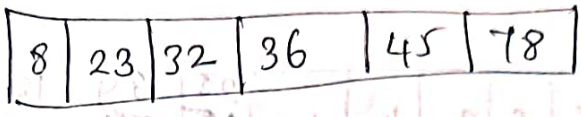
4.



5.



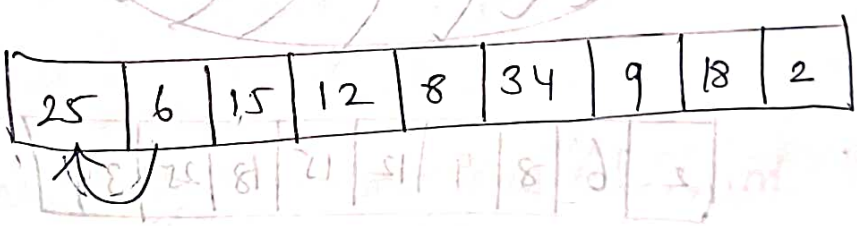
6.

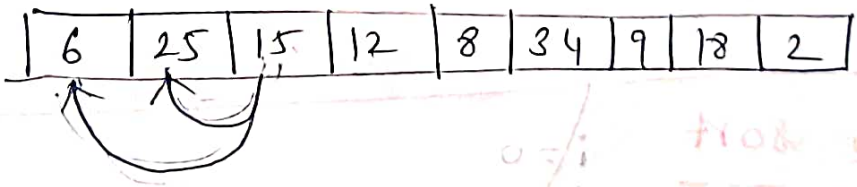


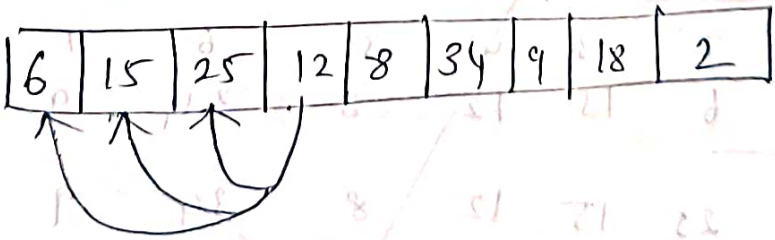
Demonstrate the insertion sort & bubble sort result for following elements.

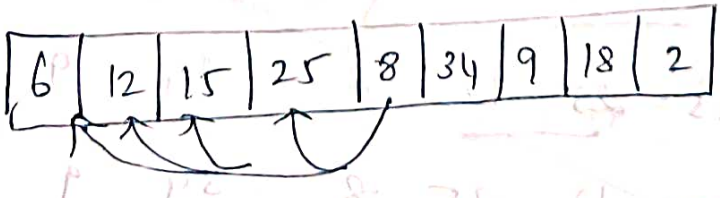
25, 6, 15, 12, 8, 34, 9, 18, 2

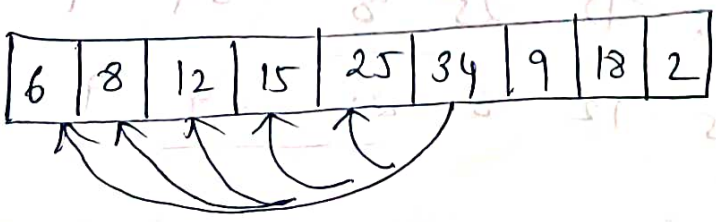
Insertion sort

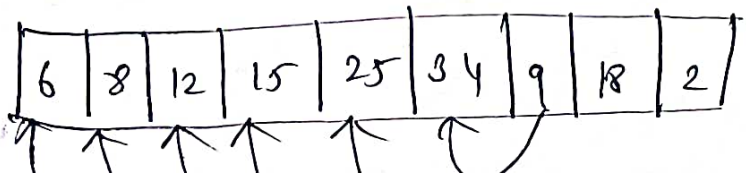
① 

② 

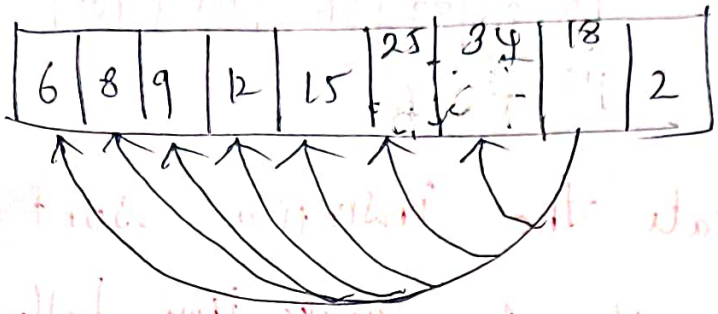
③ 

④ 

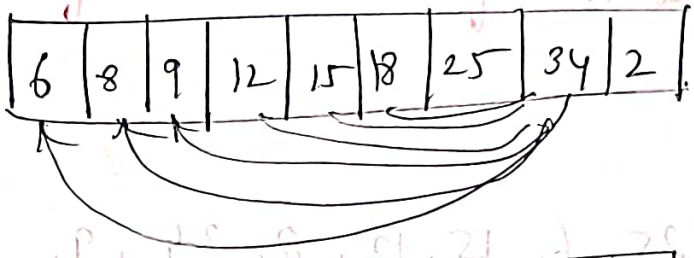
⑤ 

⑥ 

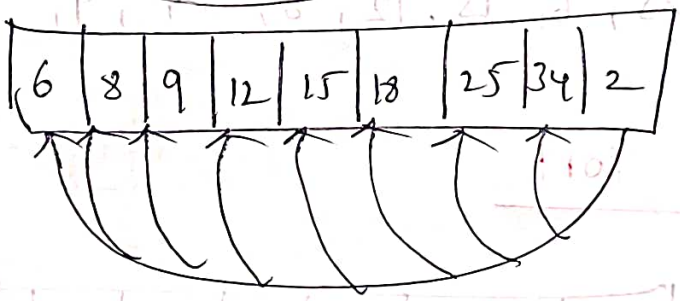
7.



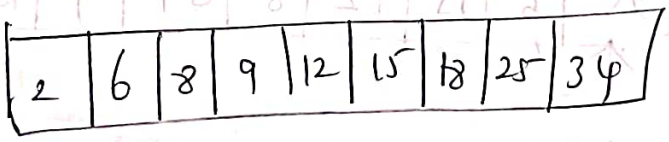
8.



9.



10.



Bubble sort

i = 0

1	2	3	4	5	6	7	8	9
25	6	15	12	8	34	9	18	2
6	25	15	12	8	34	9	18	2
6	15	25	12	8	34	9	18	2
6	15	12	25	8	34	9	18	2
6	15	12	8	25	34	9	18	2

Insertion sort after each pass

Example: 34, 8, 64, 51, 32, 21

							positions moved
After P=1	8	34	64	51	32	21	1
After P=2	8	34	64	51	32	21	0
After P=3	8	34	51	64	32	21	1
After P=4	8	32	34	51	64	21	3
After P=5	8	21	32	34	51	64	4

Insertion sort routine

```
Void InsertionSort(ElementType A[], int N)
{
    int j, P;
    int Tmp;
    for (P=1; P<N; P++)
    {
        Tmp = A[P];
        for (j=P; j>0 && A[j-1] > Tmp; j--)
            A[j] = A[j-1];
        A[j] = Tmp;
    }
}
```