WEEK-3

Date:02-07-2025

List of programs:

1. Write a C program to sort a given list of integers using Bubble Sort Technique.
2. Write a C program to sort a given list of integers using Selection Sort Technique
3. Write a C program to sort a given list of integers using Insertion Sort Technique

1. Aim: To write a C program to sort a given list of integers using Bubble Sort Technique.

Program:

```
#include<stdio.h>
int main()
{
int i,j,n,a[100],temp;
printf("enter the number of elements");
scanf("%d",&n);
printf("enter %d elements\n",n);
for (i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
for (i=0;i<n;i++)
{
for (j=0;j<n-1;j++)
{
if (a[j]>a[j+1])
{
temp= a[j];
a[j]=a[j+1];
```

```
a[j+1]=temp;
}
}
}
printf("elements after sorting:\n");
for (i=0;i<n;i++)
{
printf("%d\n",a[i]);
}
return 0;
}
```

Output:

```
enter the number of elements4
enter 4 elements
12
4
13
24
elements after sorting:
4
12
13
24


=== Code Execution Successful ===
```

2. Aim: Write a C program to sort a given list of integers using Selection Sort Technique.

Program:

```
#include<stdio.h>
int main()
{
int i,j,n,a[100],t,min,k;
printf("enter the number of elements");
scanf("%d",&n);
printf("enter %d elements\n",n);
for (i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
for (i=0;i<n;i++)
{
min=i;
for (j=i+1;j<n;j++)
{
if (a[min]>a[j])
{
min=j;
if(min!=k)
{
t=a[i];
a[i]=a[min];
```

```
a[min]=t;
}
}
}
}
printf("elements after sorting:\n");
for (i=0;i<n;i++)
{
printf("%d \n",a[i]);
}
return 0;
}
```

Output:

```
enter the number of elements5
enter 5 elements
12
23
1
13
4
elements after sorting:
4
1
13
12
23


=== Code Execution Successful ===
```

3. Aim: Write a C program to sort a given list of integers using Insertion Sort Technique.

Program:

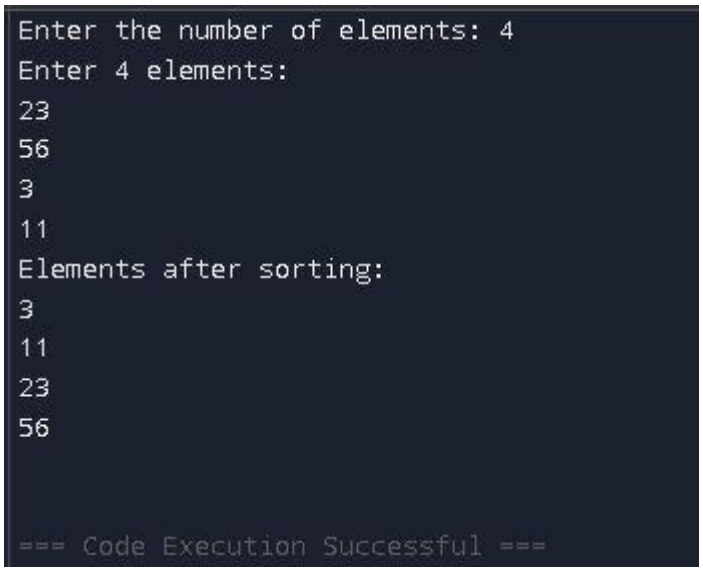
```
#include <stdio.h>
int main()
{
   int i, j, n, a[100], temp;
   printf("Enter the number of elements: ");
   scanf("%d", &n);
   printf("Enter %d elements:\n", n);
   for (i = 0; i < n; i++)
   {
      scanf("%d", &a[i]);
   }
   for (i = 1; i < n; i++)
   {
      temp = a[i];
      for (j = i - 1; j >= 0 && a[j] > temp; j--)
      {
         a[j + 1] = a[j];
      }

      a[j + 1] = temp;
   }
```

```
    printf("Elements after sorting:\n");
    for (i = 0; i < n; i++)
    {

       printf("%d\n", a[i]);
    }
    return 0;
}
```

Output:



Inferences:

- Bubble sort working principle is Repeatedly compares adjacent elements and swaps them if they are in the wrong order.

- Bubble Sort is **easy but inefficient**. It is stable, adaptive (with optimization), and good for learning or very small datasets, but not used in practice for large-scale sorting.

- Selection sort working priciple is **Working Principle** Finds the **minimum (or maximum)** element from the unsorted part of the array. Places it at the correct position by swapping with the first unsorted element. Repeats until the array is sorted.

- Selection Sort is simple and requires fewer swaps than Bubble Sort, but it is **always $O(n^2)$** and **not adaptive**. It is **inefficient for large datasets**, but sometimes used when **swapping is expensive**.

- Insertion Sort is **better than Bubble and Selection Sort** for **small or nearly sorted arrays**. It is **stable, adaptive, and simple**, but still **$O(n^2)$** for large unsorted datasets, making it impractical for large-scale sorting.