********** Unit - V **********

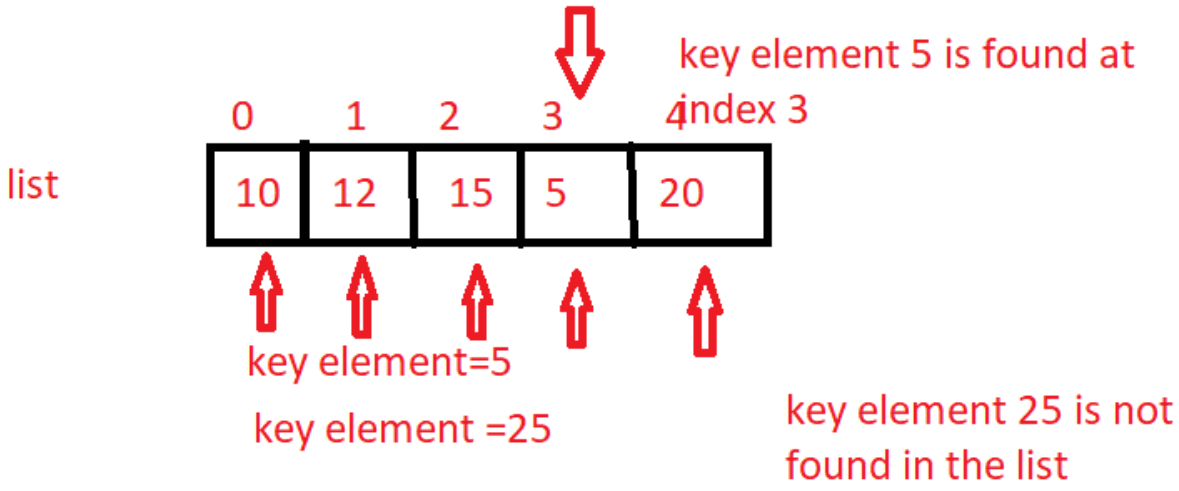
Searching
Sorting

Searching:

Finding the given element from an array is known as searching.

The following are 2 searching techniques
1. Linear Search
2. Binary Search

1. Linear Search:
In the Linear search we start comparing key element from begining of the array till the end or till the element found



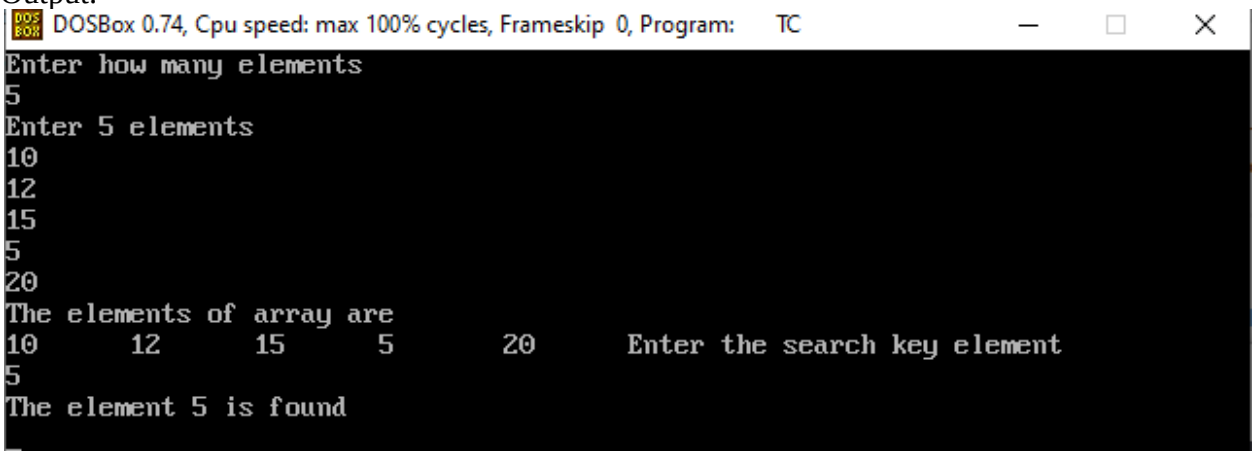
Write a program to search for an element using linear search technique.

lsearch.c

```
#include<stdio.h>
main()
{
        int list[20],i,n,skey,found=0;
        clrscr();
        printf("Enter how many elements\n");
        scanf("%d",&n);
        printf("Enter %d elements\n",n);
        for(i=0;i<n;i++)
        {
                scanf("%d",&list[i]);
```

```
        }
        printf("The elements of array are\n");
        for(i=0;i<n;i++)
        {
                printf("%d\t",list[i]);
        }
        printf("Enter the search key element\n");
        scanf("%d",&skey);
        for(i=0;i<n;i++)
        {
                if(list[i]==skey)
                {
                        found=1;
                        break;
                }
        }
        if(found==1)
        {
                printf("The element %d is found\n",skey);
        }
        else
        {
                printf("The element %d is not found\n",skey);
        }
        return 0;
}
```

Output:



2. Binary Search:
In binary search elements of array must be in sorted order.

In Binary Search we will divide the array into 2 parts based on the mid index, We compare the search element with the element at mid index, if these two elements are equal then we return the element is found, otherwise we check for the search element is less than or greater than to mid index element, if search element is less than mid index element then we change the high index to mid-1, if search element is greater than mid index element then we change the low index to mid+1. This process will repeat until the element is found or till the low value is less than or equal to high value.

n=8

	low			mid				high
	0	1	2	3	4	5	6	7
a	10	20	40	60	70	80	90	100

skey= 80
low=0
high=7
mid=(low+high)/2
(0+7)/2= 3
(4+7)/2 = 5

	low	mid		high
	4	5	6	7
	70	80	90	100

if (a[mid]==skey) 60==80 -> F
80==80 -> T

found=1

if(a[mid]<skey) 60<80 -> T
low=mid+1

if(a[mid]>skey)
high=mid-1

n=8

	low			mid				high
	0	1	2	3	4	5	6	7
a	10	20	40	60	70	80	90	100

skey= 40
low=0
high=7
mid=(low+high)/2
(0+7)/2= 3
(0+2)/2=1
(2+2)/2=2

	low	mid	high
	0	1	2
	10	20	40

	mid
	low
	high
	2
	40

if (a[mid]==skey)	60==40 -> F
	20==40 -> F
found=1	40==40 -> T
if(a[mid]<skey)	60<40 -> F
low=mid+1	20<40 -> T
if(a[mid]>skey)	60>40 -> T
high=mid-1	

Write a program to search for an element from the array using binary search technique.

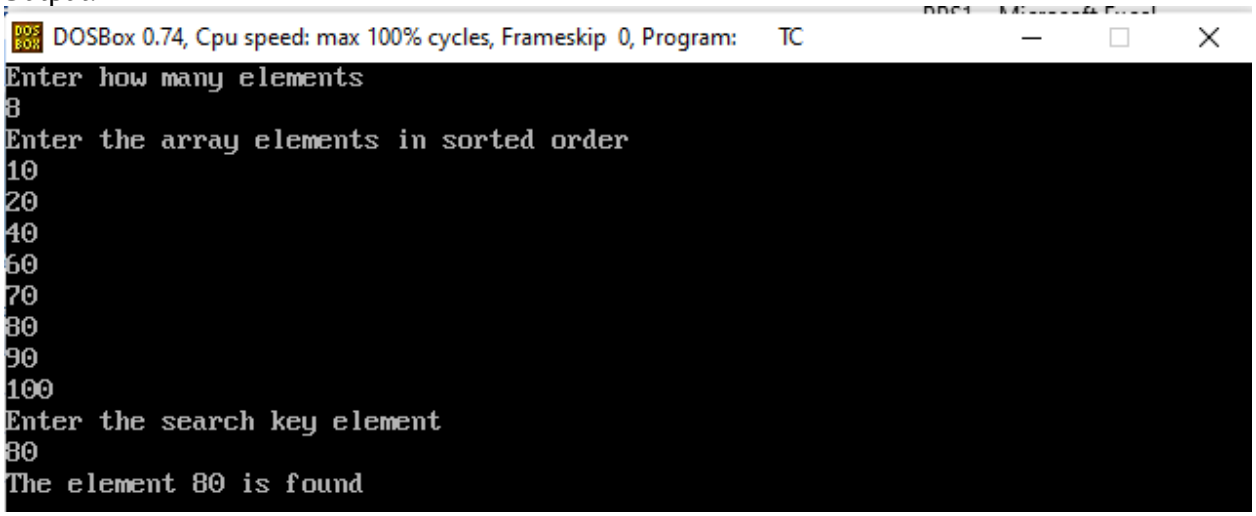
bsearch.c

```
#include<stdio.h>
main()
{
        int a[50],n,i,mid,low,high,found=0,skey;
        clrscr();
        printf("Enter how many elements \n");
        scanf("%d",&n);
        printf("Enter the array elements in sorted order\n");
        for(i=0;i<n;i++)
        {
```

```
        scanf("%d",&a[i]);
    }
    printf("Enter the search key element\n");
    scanf("%d",&skey);

    low=0;
    high=n-1;
    while(low<=high)
    {
        mid=(low+high)/2;
        if(a[mid]==skey)
        {
            found=1;
            break;
        }
        else if(a[mid]<skey)
        {
            low=mid+1;
        }
        else
        {
            high=mid-1;
        }
    }
    if(found==1)
        printf("The element %d is found\n",skey);
    else
        printf("The element %d is not found\n",skey);
    return 0;
}
```

Output:



Sorting

Arranging the elements of array either in ascending or in descending order is known sorting.

The following are the various sorting techniques
1. Bubble Sort
2. Selection Sort
3. Insertion Sort
4. Merge Sort
5. Quick Sort

1. Bubble Sort:

In bubble sort technique we sort the elements of array by comparing with neighborhood elements. The comparison process will repeat until all the elements of array are adjusted in the sorted order.

n=8

	0	1	2	3	4	5	6	7
a	6	5	3	1	8	7	2	4

Iteration 1

6 > 5 -> T
swap

	0	1	2	3	4	5	6	7
a	5	6	3	1	8	7	2	4

6 > 3 -> T
swap

	0	1	2	3	4	5	6	7
a	5	3	6	1	8	7	2	4

6 > 1 -> T
swap

	0	1	2	3	4	5	6	7
a	5	3	1	6	8	7	2	4

6 > 8 -> F No swap

	0	1	2	3	4	5	6	7
a	5	3	1	6	8	7	2	4

8 > 7 -> T

swap

	0	1	2	3	4	5	6	7
a	5	3	1	6	7	8	2	4

8 > 2 -> T
swap

	0	1	2	3	4	5	6	7
a	5	3	1	6	7	2	8	4

8 > 4 -> T
swap

	0	1	2	3	4	5	6	7
a	5	3	1	6	7	2	4	8

Iteration 2

	0	1	2	3	4	5	6	7
a	5	3	1	6	7	2	4	8

5 > 3 -> T
swap

	0	1	2	3	4	5	6	7
a	3	5	1	6	7	2	4	8

5 > 1 -> T
swap

	0	1	2	3	4	5	6	7
a	3	1	5	6	7	2	4	8

5 > 6 -> F No swap

	0	1	2	3	4	5	6	7
a	3	1	5	6	7	2	4	8

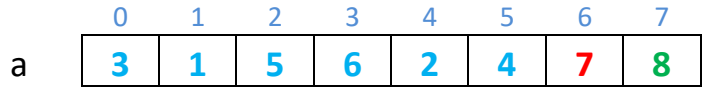
6 > 7 -> F No swap

	0	1	2	3	4	5	6	7
a	3	1	5	6	7	2	4	8

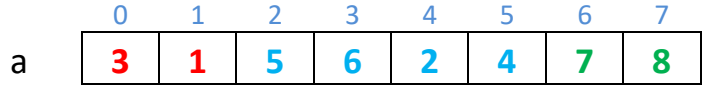
7 > 2 -> T
swap

	0	1	2	3	4	5	6	7
a	3	1	5	6	2	7	4	8

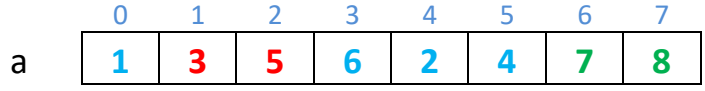
7 > 4 -> T
swap



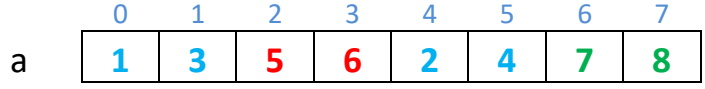
Iteration 3



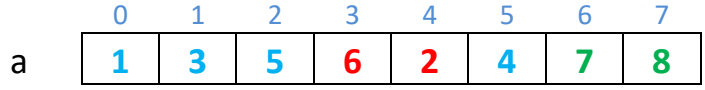
3 > 1 -> T
swap



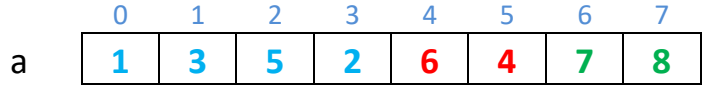
3 > 5 -> F No swap



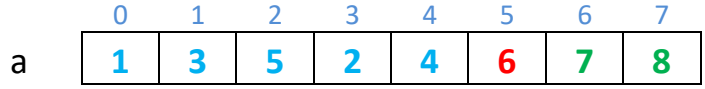
5 > 6 -> F No swap



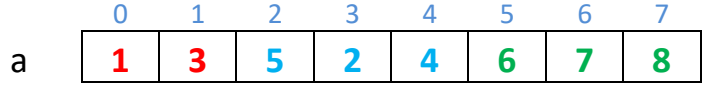
6 > 2 -> T
swap



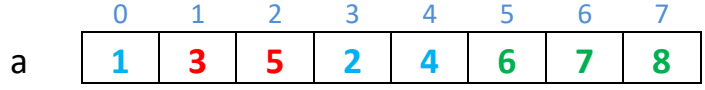
6 > 4 -> T
swap



Iteration 4



1 > 3 -> F No swap



3 > 5 -> F No swap

a

0	1	2	3	4	5	6	7
1	3	5	2	4	6	7	8

5 > 2 -> T
swap

a

0	1	2	3	4	5	6	7
1	3	2	5	4	6	7	8

5 > 4 -> T
swap

a

0	1	2	3	4	5	6	7
1	3	2	4	5	6	7	8

Iteration 5

a

0	1	2	3	4	5	6	7
1	3	2	4	5	6	7	8

1 > 3 -> F No swap

a

0	1	2	3	4	5	6	7
1	3	2	4	5	6	7	8

3 > 2 -> T
swap

a

0	1	2	3	4	5	6	7
1	2	3	4	5	6	7	8

3 > 4 -> F No swap

Iteration 6

a

0	1	2	3	4	5	6	7
1	2	3	4	5	6	7	8

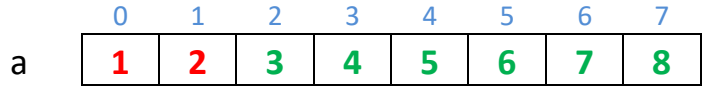
1 > 2 -> F No swap

a

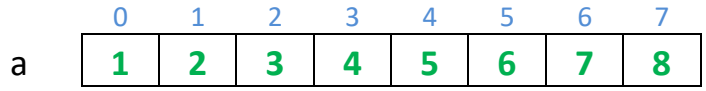
0	1	2	3	4	5	6	7
1	2	3	4	5	6	7	8

2 > 3 -> F No swap

Iteration 7



1 > 2 -> F No swap



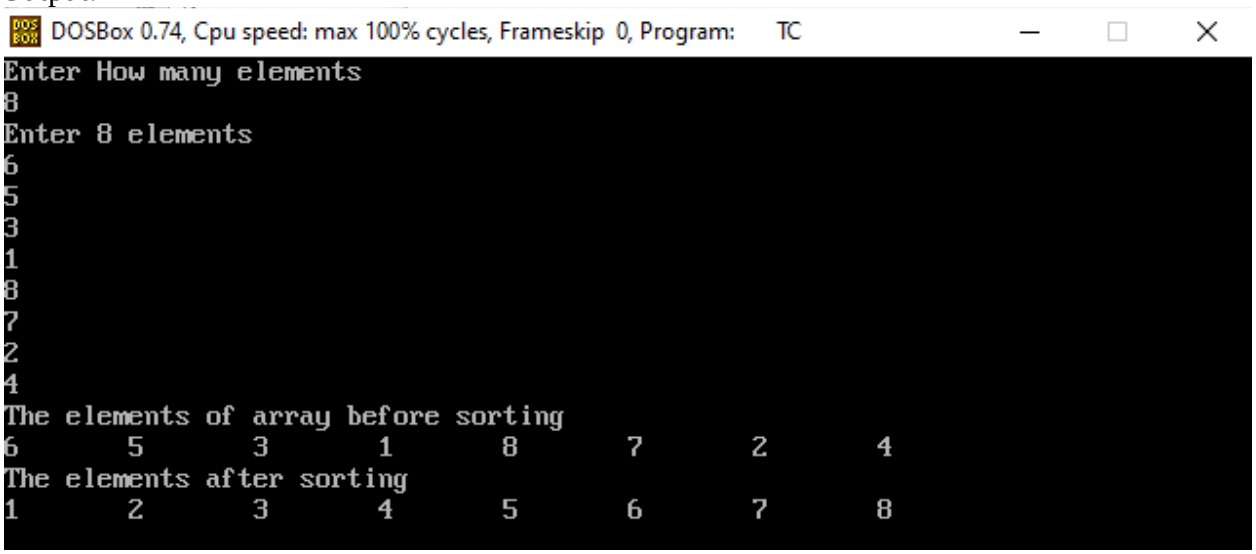
Write a program to sort the elements of array using bubble sort technique

bubble.c

```
#include<stdio.h>
main()
{
	int a[50],n,i,j,temp;
	clrscr();
	printf("Enter How many elements\n");
	scanf("%d",&n);
	printf("Enter %d elements\n",n);
	for(i=0;i<n;i++)
	{
		scanf("%d",&a[i]);
	}
	printf("The elements of array before sorting\n");
	for(i=0;i<n;i++)
	{
		printf("%d\t",a[i]);
	}
	for(i=0;i<n-1;i++)
	{
		for(j=0;j<n-1-i;j++)
		{
			if(a[j]>a[j+1])
			{
				temp=a[j];
				a[j]=a[j+1];
				a[j+1]=temp;
			}
		}
```

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

Output:



2. Selection Sort:

In this technique we select a location and place the correct element in the selected location.

n=8

	0	1	2	3	4	5	6	7
a	6	5	3	1	8	7	2	4

Iteration 1

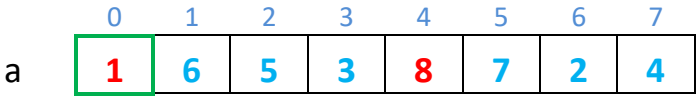
6 > 5 -> T swap

	0	1	2	3	4	5	6	7
a	5	6	3	1	8	7	2	4

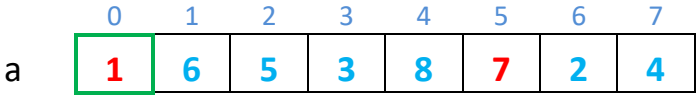
5 > 3 -> T swap

	0	1	2	3	4	5	6	7
a	3	6	5	1	8	7	2	4

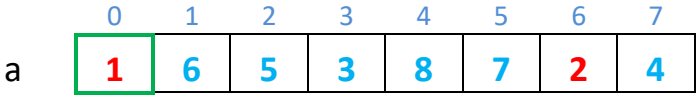
3 > 1 -> T swap



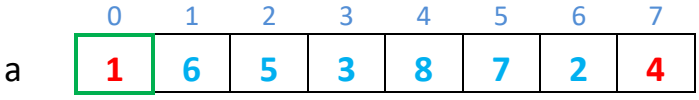
1 > 8 -> F No swap



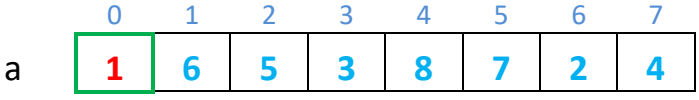
1 > 7 -> F No swap



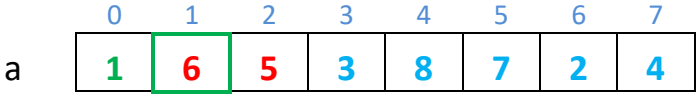
1 > 2 -> F No swap



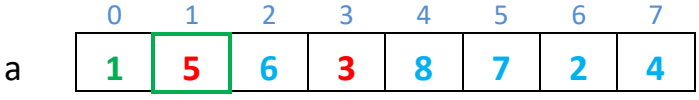
1 > 4 -> F No swap



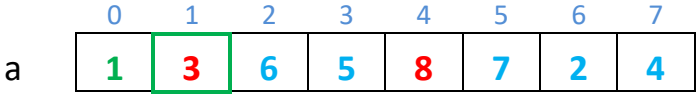
Iteration 2



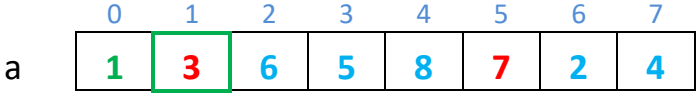
6 > 5 -> T swap



5 > 3 -> T swap



3 > 8 -> F No swap



3 > 7 -> F No swap

a

0	1	2	3	4	5	6	7
1	3	6	5	8	7	2	4

3 > 2 -> T swap

a

0	1	2	3	4	5	6	7
1	2	6	5	8	7	3	4

2 > 4 -> F No swap

Iteration 3

a

0	1	2	3	4	5	6	7
1	2	6	5	8	7	3	4

6 > 5 -> T swap

a

0	1	2	3	4	5	6	7
1	2	5	6	8	7	3	4

5 > 8 -> F No swap

a

0	1	2	3	4	5	6	7
1	2	5	6	8	7	3	4

5 > 7 -> F No swap

a

0	1	2	3	4	5	6	7
1	2	5	6	8	7	3	4

5 > 3 -> T swap

a

0	1	2	3	4	5	6	7
1	2	3	6	8	7	5	4

3 > 4 -> F No swap

a

0	1	2	3	4	5	6	7
1	2	3	6	8	7	5	4

Iteration 4

a

0	1	2	3	4	5	6	7
1	2	3	6	8	7	5	4

6 > 8 -> F No swap

a

0	1	2	3	4	5	6	7
1	2	3	6	8	7	5	4

6 > 7 -> F No swap

a

0	1	2	3	4	5	6	7
1	2	3	6	8	7	5	4

6 > 5 -> T swap

a

0	1	2	3	4	5	6	7
1	2	3	5	8	7	6	4

5 > 4 -> T swap

a

0	1	2	3	4	5	6	7
1	2	3	4	8	7	6	5

Iteration 5

a

0	1	2	3	4	5	6	7
1	2	3	4	8	7	6	5

8 > 7 -> T swap

a

0	1	2	3	4	5	6	7
1	2	3	4	7	8	6	5

7 > 6 -> T swap

a

0	1	2	3	4	5	6	7
1	2	3	4	6	8	7	5

6 > 5 -> T swap

a

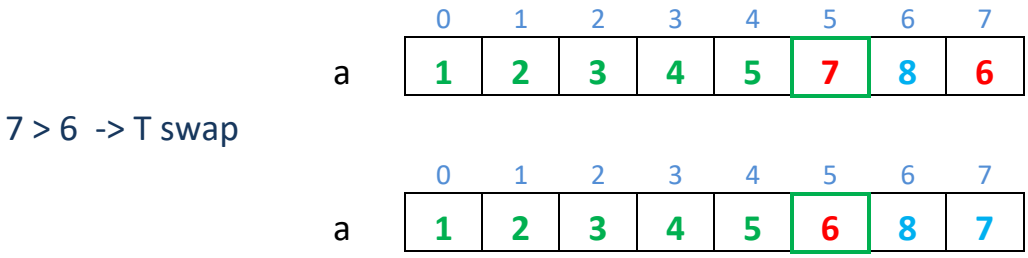
0	1	2	3	4	5	6	7
1	2	3	4	5	8	7	6

Iteration 6

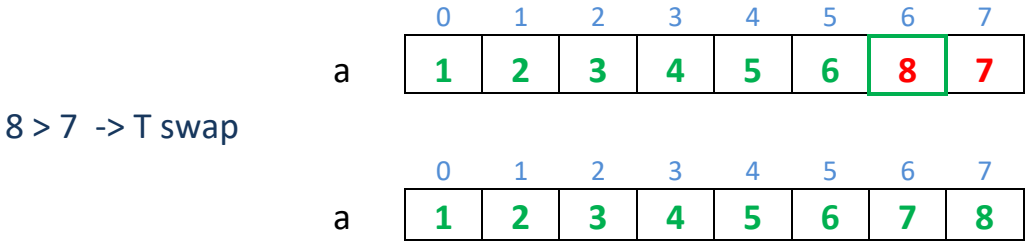
a

0	1	2	3	4	5	6	7
1	2	3	4	5	8	7	6

8 > 7 -> T swap



Iteration 7



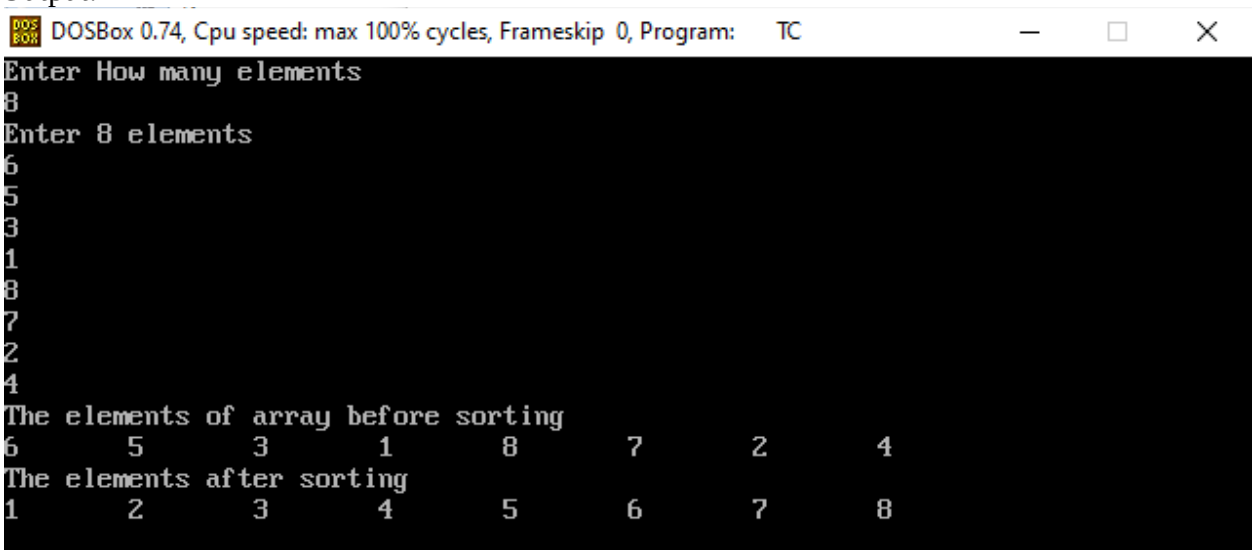
Write a program to sort the elements of array using selection sort technique

selection.c

```
#include<stdio.h>
main()
{
	int a[50],n,i,j,temp;
	clrscr();
	printf("Enter How many elements\n");
	scanf("%d",&n);
	printf("Enter %d elements\n",n);
	for(i=0;i<n;i++)
	{
		scanf("%d",&a[i]);
	}
	printf("The elements of array before sorting\n");
	for(i=0;i<n;i++)
	{
		printf("%d\t",a[i]);
	}
	for(i=0;i<n-1;i++)
	{
		for(j=i+1;j<n;j++)
		{
			if(a[i]>a[j])
			{
```

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

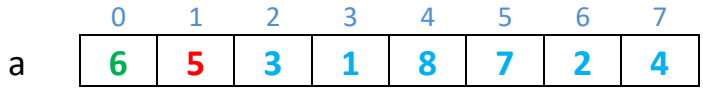
Output:



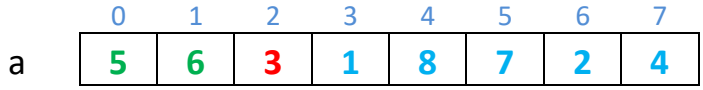
3. Insertion Sort:

In this technique we divide the list as sorted list and unsorted list, then select an element from the unsorted list and place the element in the correct location of the sorted list. This process will repeat until all the element comes into sorted list.

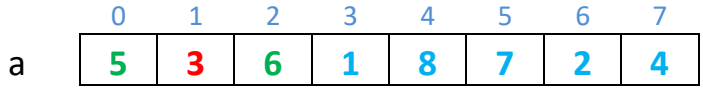
n=8



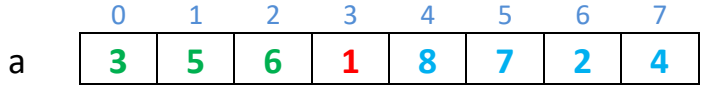
6 > 5 -> T swap



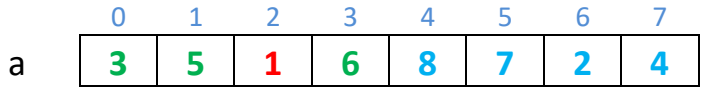
6 > 3 -> T swap



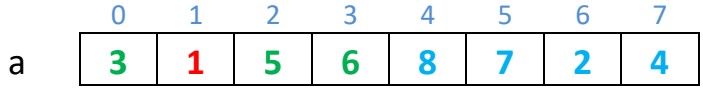
5 > 3 -> T swap



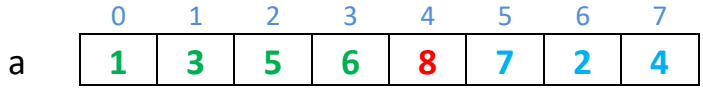
6 > 1 -> T swap



5 > 1 -> T swap

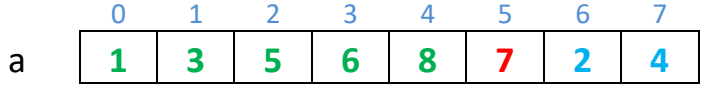


3 > 1 -> T swap



6 > 8 -> F No swap

Move to Next Element



8 > 7 -> T swap

a

0	1	2	3	4	5	6	7
1	3	5	6	7	8	2	4

6 > 7 -> F No swap
Move to Next Element

a

0	1	2	3	4	5	6	7
1	3	5	6	7	8	2	4

8 > 2 -> T swap

a

0	1	2	3	4	5	6	7
1	3	5	6	7	2	8	4

7 > 2 -> T swap

a

0	1	2	3	4	5	6	7
1	3	5	6	2	7	8	4

6 > 2 -> T swap

a

0	1	2	3	4	5	6	7
1	3	5	2	6	7	8	4

5 > 2 -> T swap

a

0	1	2	3	4	5	6	7
1	3	2	5	6	7	8	4

3 > 2 -> T swap

a

0	1	2	3	4	5	6	7
1	2	3	5	6	7	8	4

1 > 2 -> F No swap
Move to Next Element

a

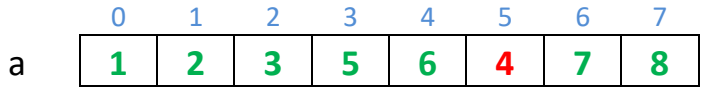
0	1	2	3	4	5	6	7
1	2	3	5	6	7	8	4

8 > 4 -> T swap

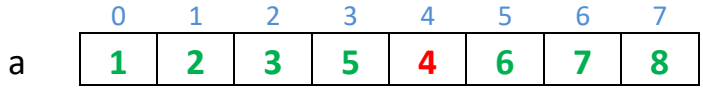
a

0	1	2	3	4	5	6	7
1	2	3	5	6	7	4	8

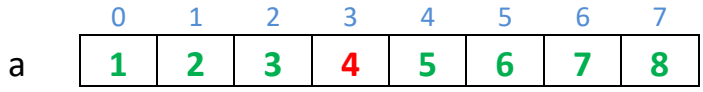
7 > 4 -> T swap



6 > 4 -> T swap

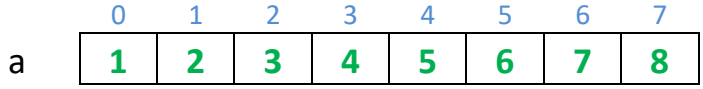


5 > 4 -> T swap



3 > 4 -> F No swap

Move to Next Element



If no next element sorting is done

Write a program to sort the elements of array using Insertion sort technique

insertion.c

```
#include<stdio.h>
main()
{
	int a[50],n,i,j,temp;
	clrscr();
	printf("Enter How many elements\n");
	scanf("%d",&n);
	printf("Enter %d elements\n",n);
	for(i=0;i<n;i++)
	{
		scanf("%d",&a[i]);
	}
	printf("The elements of array before sorting\n");
	for(i=0;i<n;i++)
	{
		printf("%d\t",a[i]);
	}
```

```
        i=1;
        while(i<n)
        {
                j=i;
                while( (j>0) && (a[j-1]>a[j]) )
                {
                        temp=a[j];
                        a[j]=a[j-1];
                        a[j-1]=temp;
                        j--;
                }
                i++;
        }


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

Output: