Pranjal Save(Sy-comps)

DATA STRUCTURE USING ‘C’(22317)

UNIT II : SEARCHING AND SORTING

(Weightage-12marks)

SYLLABUS;

SEARCHING – Linear search and Binary search

SORTING- Quick sort(**Option**) , Bubble sort , Insertion sort , Selection sort , Radix sort

---

Q1. State the terms – Searching & Sorting (2marks)

Answer-

Searching

- Searching is the process of finding some particular element in the list.
- If the element is present in the list, then the process is called successful, and the process returns the location of that element; otherwise, the search is called unsuccessful.
- Two popular search methods are
- Linear Search * Binary Search.

Sorting

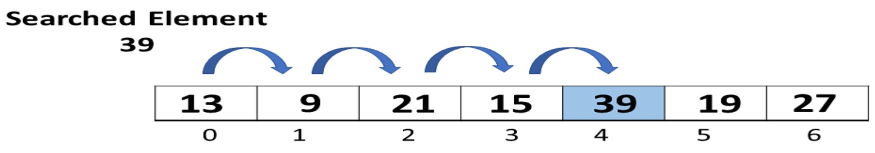
- A Sorting Algorithm is used to rearrange a given array or list elements according to a comparison operator on the elements.
- The comparison operator is used to decide the new order of elements in the respective data structure. Following are sorting techniques:
- Selection sort ,Bubble sort ,Insertion sort , Quick sort , Radix sort

---

Q2. Describe working of Linear search with an suitable algorithm. (4marks)

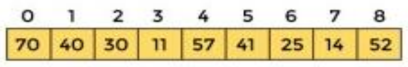
Answer –

- Linear search is also called as sequential search algorithm.
- It is the simplest searching algorithm.
- In Linear search, we simply traverse the list completely and match each element of the list with the item whose location is to be found.
- If the match is found, then the location of the item is returned; otherwise, the algorithm returns NULL.
- The method in which the elements to be searched is checked in entire data structure in a sequential way from starting to end is known as Linear search.
- Linear search is an easiest and straightmost searching technique.

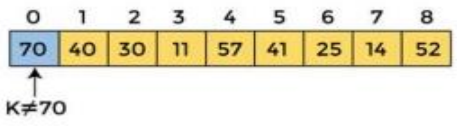


Working-
Now, let's see the working of the linear search Algorithm. To understand the working of linear search algorithm, let's take an unsorted array. It will be easy to understand the working of linear search with an example

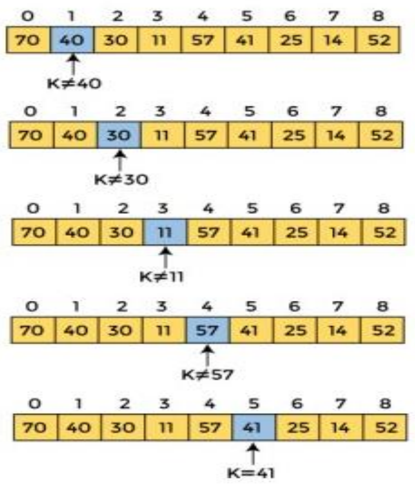
Let the elements of array are -



Let the element to be searched is **K = 41**
Now, start from the first element and compare **K** with each element of the array.



The value of **K**, i.e., **41**, is not matched with the first element of the array. So, move to the next element. And follow the same process until the respective element is found.



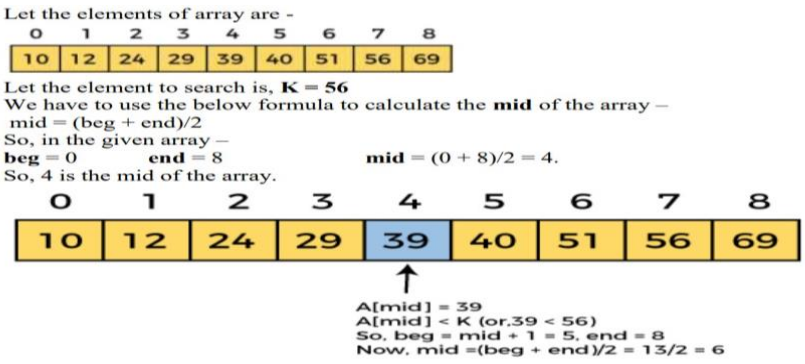
Now, the element to be searched is found. So algorithm will return the index of the element matched.

Q3. Describe Working of Binary Search with suitable example. (4marks)

- Binary search is the search technique that works efficiently on sorted lists.
- Hence, to search an element into some list using the binary search technique, we must ensure that the list is sorted.
- Binary search follows the divide and conquer approach in which the list is divided into two halves, and the item is compared with the middle element of the list.
- If the match is found then, the location of the middle element is returned.
- Otherwise, we search into either of the halves depending upon the result produced through the match.
- There are two methods to implement the binary search algorithm –
- Iterative method ‘
- Recursive method
- The recursive method of binary search follows the divide and conquer approach

- ***Binary Search*** *is defined as a searching algorithm used in a sorted array by* ***repeatedly dividing the search interval in half****. The idea of binary search is to use the information that the array is sorted and reduce the time complexity to O(log N).*

- ***Binary Search*** *is defined as a* *searching algorithm* *used in a sorted array by* ***repeatedly dividing the search interval in half****. The idea of binary search is to use the information that the array is sorted and reduce the time complexity to O(log N).*

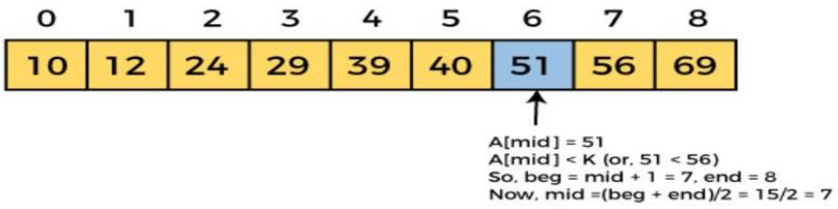


Let the elements of array are -

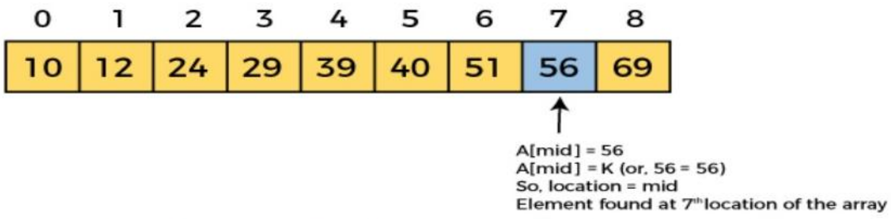
0	1	2	3	4	5	6	7	8
10	12	24	29	39	40	51	56	69

Let the element to search is, **K** = **56**
We have to use the below formula to calculate the **mid** of the array –
mid = (beg + end)/2
So, in the given array –
beg = 0 **end** = 8 **mid** = (0 + 8)/2 = 4.
So, 4 is the mid of the array.

A[mid] = 39
A[mid] < K (or,39 < 56)
So, beg = mid + 1 = 5, end = 8
Now, mid =(beg + end)/2 = 13/2 = 6



A[mid] = 51
A[mid] < K (or, 51 < 56)
So, beg = mid + 1 = 7, end = 8
Now, mid =(beg + end)/2 = 15/2 = 7



A[mid] = 56
A[mid] = K (or, 56 = 56)
So, location = mid
Element found at 7th location of the array

Now, the element to search is found. So algorithm will return the index of the element matched.

Q4. Difference Between Binary and Linear Seach (4marks)

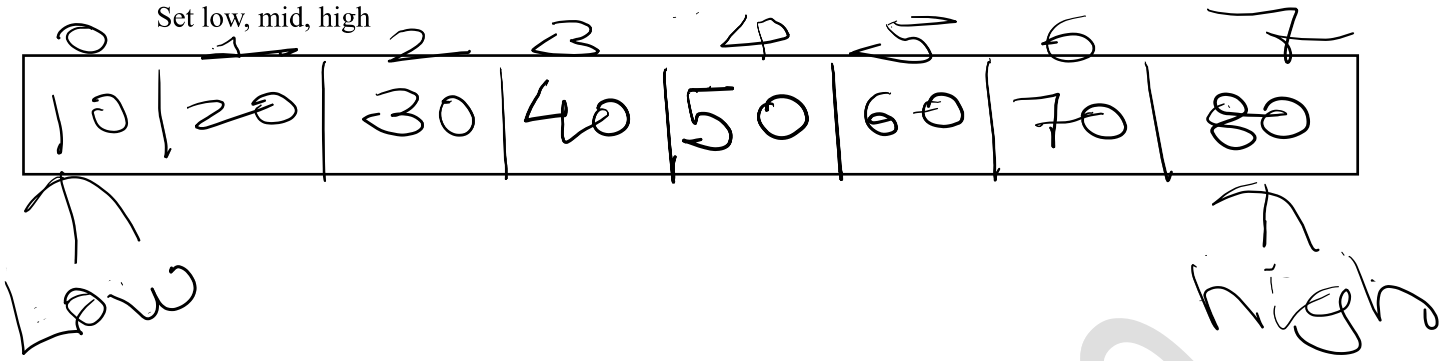
Sr. No.	Binary Search	Sequential search (linear search)
1	Input data needs to be sorted in Binary Search	Input data need not to be sorted in Linear Search.
2	In contrast, binary search compares key value with the middle element of an array and if comparison is unsuccessful then cuts down search to half.	A linear search scans one item at a time, without jumping to any item.
3	Binary search implements divide and conquer approach.	Linear search uses sequential approach.
4	In binary search the worst case complexity is O(log n) comparisons.	In linear search, the worst case complexity is O(n), comparisons.
5	Binary search is efficient for the larger array.	Linear search is efficient for the smaller array.

Pranjal Save(Sy-comps)

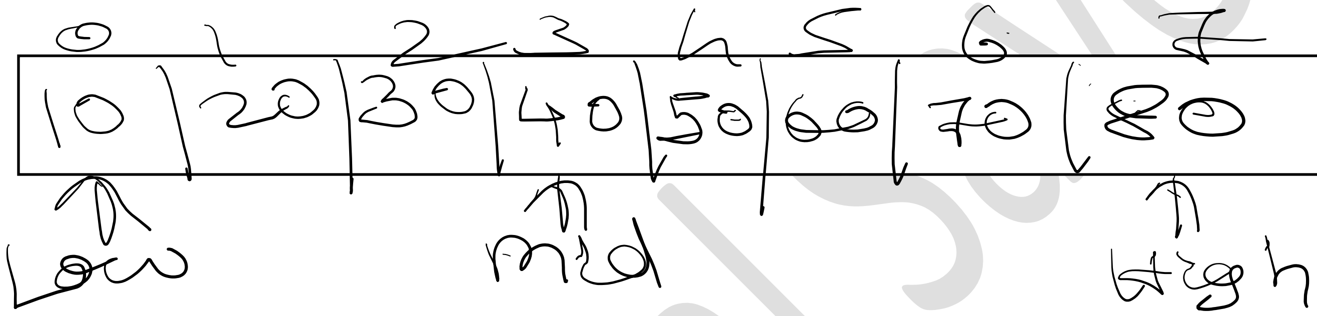
Q6. Find the location of element 20 by using Binary search algorithm .(4marks)

10,20,30,40,50,60,70,80,90

Set low, mid, high



Low=0 , Hight=7, MID=low+high/2 = 7/2 = 3

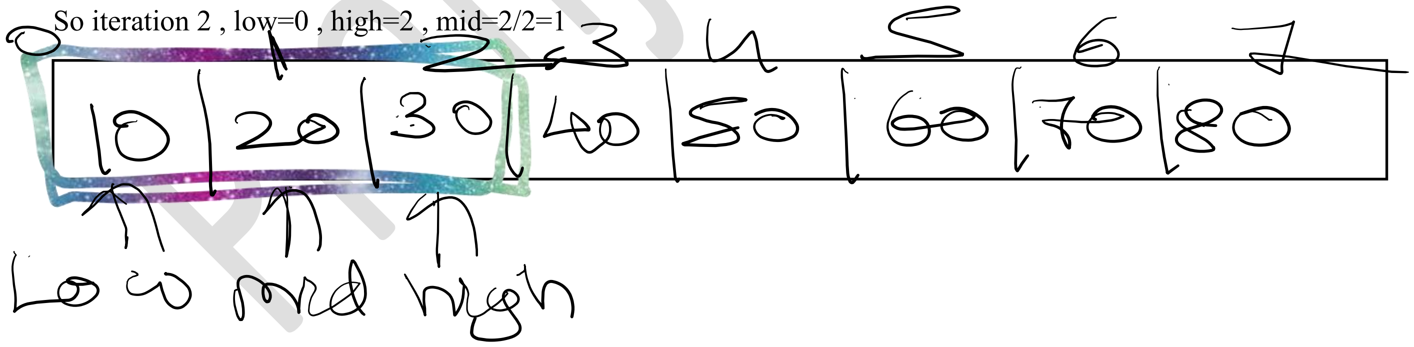


Compare 20 with the mids Element (mid element=40)

20<40

Now according to condition, High=mid-1

So iteration 2 , low=0 , high=2 , mid=2/2=1



Compare 20 with mid element (Mid element=20)

20=20

So therefore condition meet.Print its index number

Element found at index number 2.

Pranjal Save(Sy-comps)

Q7. Find the location of element 30 by using Binary search algorithm. (4marks)

10,5,20,25,8,30,40

1. Start with the entire sorted list: 5, 8, 10, 20, 25, 30, 40
2. Define two pointers, `left` and `right`, initially pointing to the first and last elements of the list: left = 0 right = 6
3. Calculate the middle index between `left` and `right`: middle = (left + right) // 2 = (0 + 6) // 2 = 3
4. Compare the middle element (list[middle]) with the target element (30):
    - If list[middle] is equal to 30, you have found the element, and its location is the middle index (3).
    - If list[middle] is greater than 30, update `right` to `middle - 1` to search in the left half of the list.
    - If list[middle] is less than 30, update `left` to `middle + 1` to search in the right half of the list.

In this case, list[3] is 20, which is less than 30. So, you update `left` to `middle + 1`, which becomes 4. Now, your search range is from index 4 to index 6:

5. Repeat steps 3-4 with the new search range:
    - Calculate the new middle: middle = (left + right) // 2 = (4 + 6) // 2 = 5
    - Compare list[middle] (list[5]) with 30.

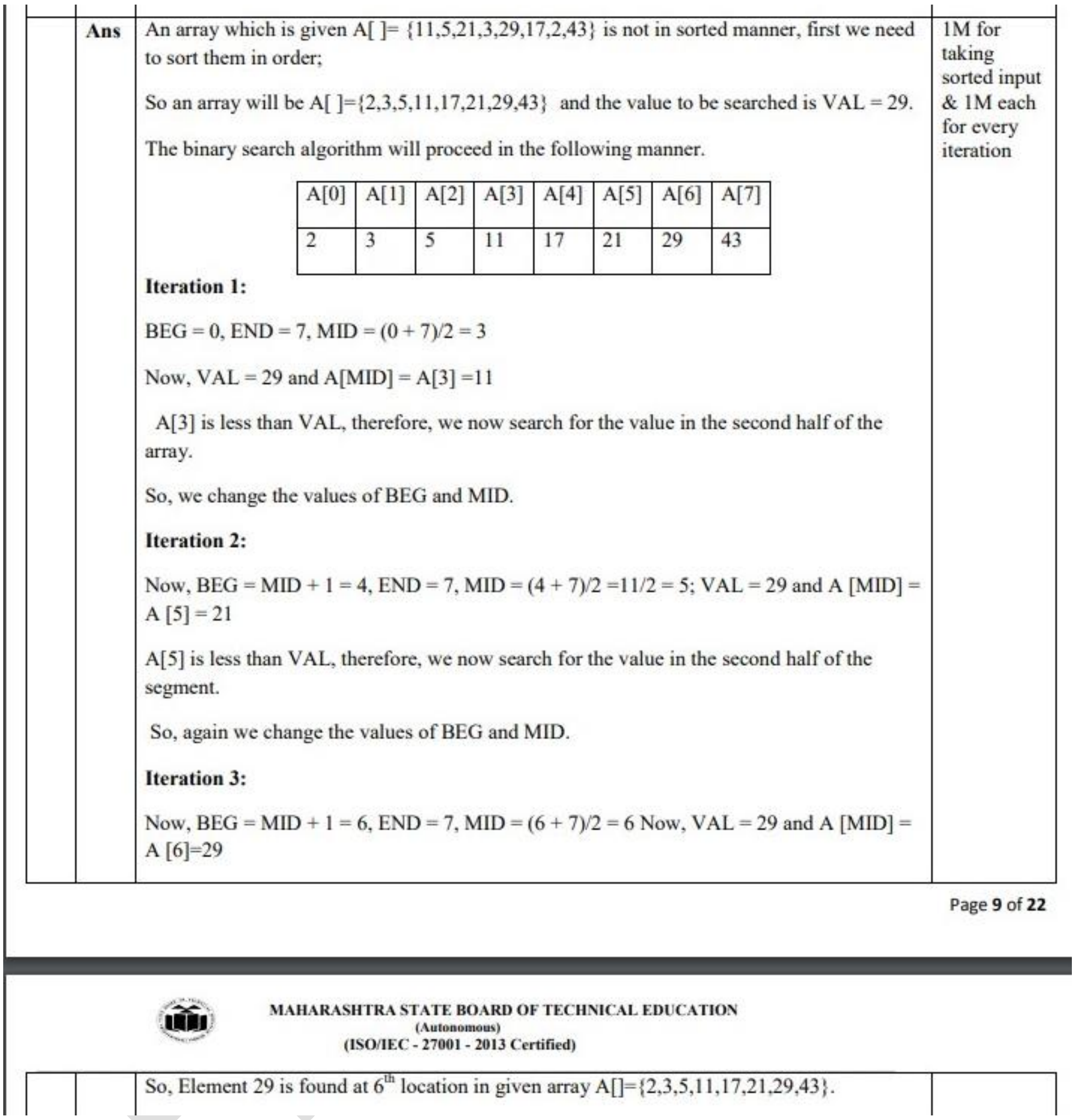
Now, list[5] is 30, which is equal to the target element. So, you have found the element, and its location is the middle index, which is 5.

So, the location of element 30 in the given list is at index 5.

Pranjal Save(Sy-comps)

Q8. Find the location of element 21 by using Binary search algorithm. (4marks)

11,5,21,3,29,17,2,45



Ans

An array which is given A[ ]= {11,5,21,3,29,17,2,43} is not in sorted manner, first we need to sort them in order;

So an array will be A[ ]={2,3,5,11,17,21,29,43} and the value to be searched is VAL = 29.

The binary search algorithm will proceed in the following manner.

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]
2	3	5	11	17	21	29	43

Iteration 1:

BEG = 0, END = 7, MID = (0 + 7)/2 = 3

Now, VAL = 29 and A[MID] = A[3] =11

A[3] is less than VAL, therefore, we now search for the value in the second half of the array.

So, we change the values of BEG and MID.

Iteration 2:

Now, BEG = MID + 1 = 4, END = 7, MID = (4 + 7)/2 =11/2 = 5; VAL = 29 and A [MID] = A [5] = 21

A[5] is less than VAL, therefore, we now search for the value in the second half of the segment.

So, again we change the values of BEG and MID.

Iteration 3:

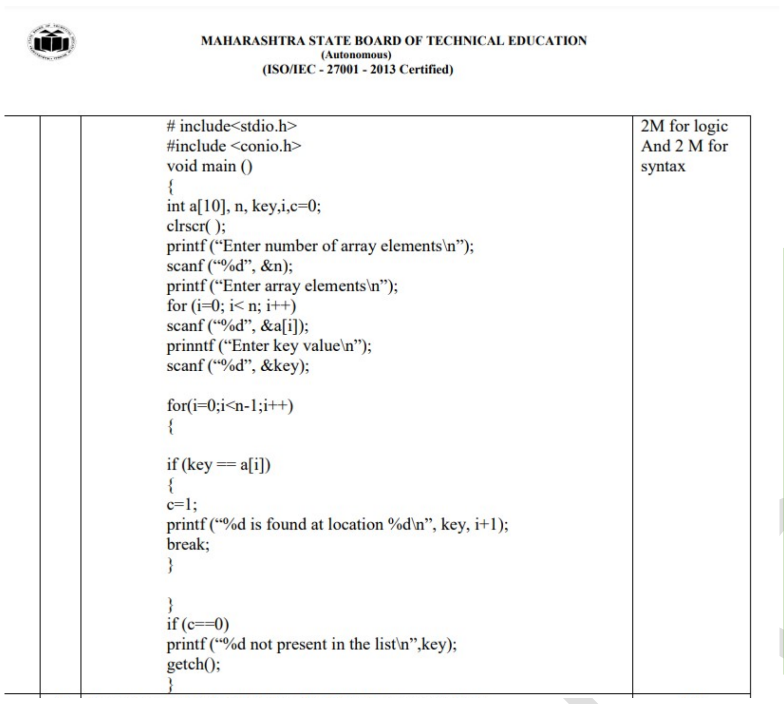
Now, BEG = MID + 1 = 6, END = 7, MID = (6 + 7)/2 = 6 Now, VAL = 29 and A [MID] = A [6]=29

1M for taking sorted input & 1M each for every iteration

So, Element 29 is found at 6th location in given array A[]={2,3,5,11,17,21,29,43}.

Pranjal Save(Sy-comps)

Q9. Write an C program for implementation of linear search. (4marks)

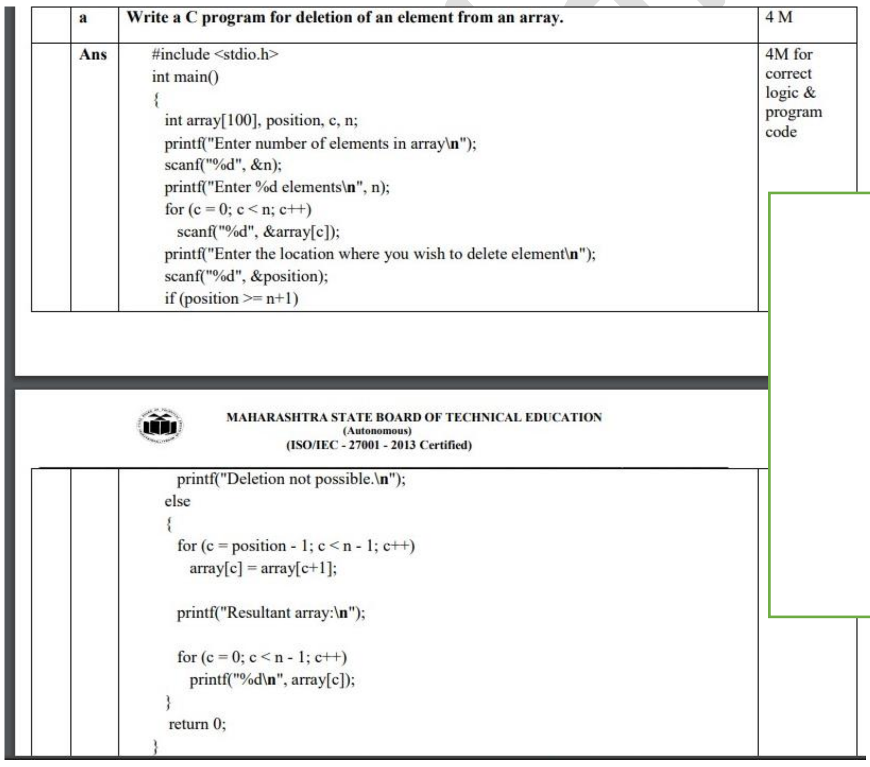


MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION
(Autonomous)
(ISO/IEC - 27001 - 2013 Certified)

```
# include<stdio.h>
#include <conio.h>
void main ()
{
int a[10], n, key,i,c=0;
clrscr( );
printf (“Enter number of array elements\n”);
scanf (“%d”, &n);
printf (“Enter array elements\n”);
for (i=0; i< n; i++)
scanf (“%d”, &a[i]);
prinntf (“Enter key value\n”);
scanf (“%d”, &key);

for(i=0;i<n-1;i++)
{

if (key == a[i])
{
c=1;
printf (“%d is found at location %d\n”, key, i+1);
break;
}

}
if (c==0)
printf (“%d not present in the list\n”,key);
getch();
}
```

2M for logic And 2 M for syntax

Q10. C Program for deletion of element of array. (4marks)



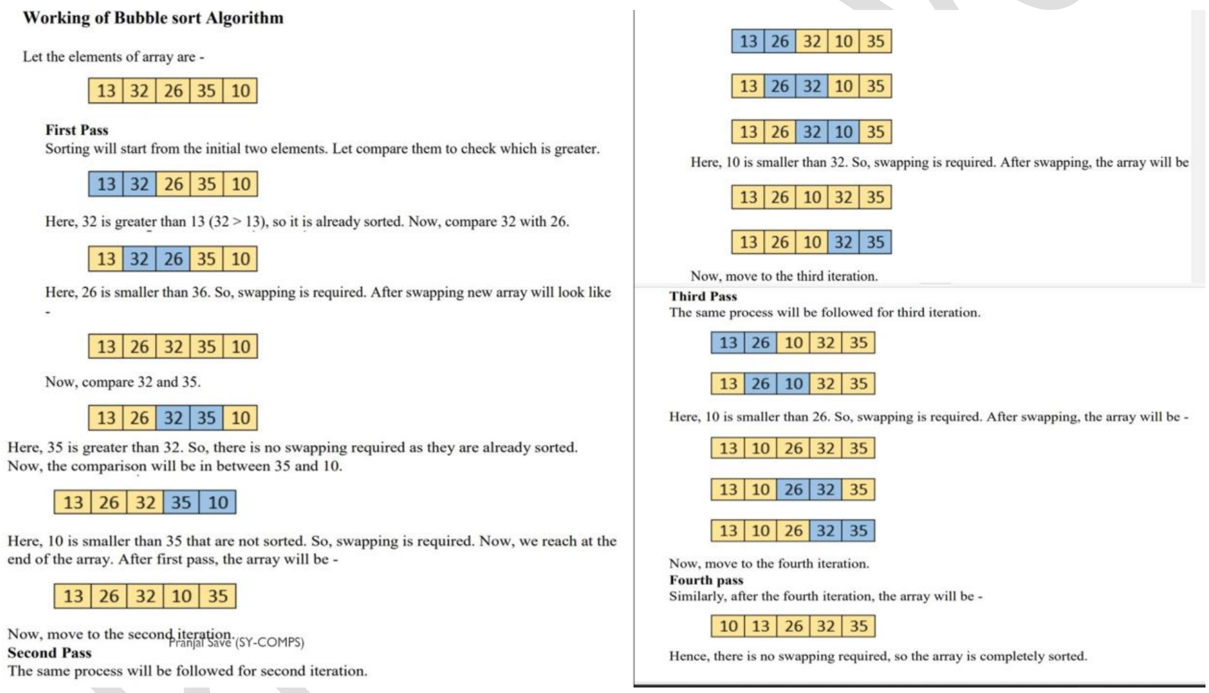
	a	Write a C program for deletion of an element from an array.	4 M
	Ans	(program below)	4M for correct logic & program code

MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION
(Autonomous)
(ISO/IEC - 27001 - 2013 Certified)

```
#include <stdio.h>
int main()
{
  int array[100], position, c, n;
  printf("Enter number of elements in array\n");
  scanf("%d", &n);
  printf("Enter %d elements\n", n);
  for (c = 0; c < n; c++)
    scanf("%d", &array[c]);
  printf("Enter the location where you wish to delete element\n");
  scanf("%d", &position);
  if (position >= n+1)
    printf("Deletion not possible.\n");
  else
  {
    for (c = position - 1; c < n - 1; c++)
      array[c] = array[c+1];

    printf("Resultant array:\n");

    for (c = 0; c < n - 1; c++)
      printf("%d\n", array[c]);
  }
  return 0;
}
```

Q11.Explain working of Bubble sort with suitable example.(4marks)

- Bubble sort works on the repeatedly swapping of adjacent elements until they are not in the intended order.
- It is called bubble sort because the movement of array elements is just like the movement of air bubbles in the water.
- Bubbles in water rise up to the surface; similarly, the array elements in bubble sort move to the end in each iteration.
- Although it is simple to use, it is primarily used as an educational tool because the performance of bubble sort is poor in the real world.
- It is not suitable for large data sets.
- The average and worst-case complexity of Bubble sort is O(n2), where n is a number of items.
- Bubble short is majorly used where –
  complexity does not matter
  simple and shortcode is preferred



Working of Bubble sort Algorithm

Let the elements of array are -

13 | 32 | 26 | 35 | 10

First Pass
Sorting will start from the initial two elements. Let compare them to check which is greater.

13 | 32 | 26 | 35 | 10

Here, 32 is greater than 13 (32 > 13), so it is already sorted. Now, compare 32 with 26.

13 | 32 | 26 | 35 | 10

Here, 26 is smaller than 36. So, swapping is required. After swapping new array will look like -

13 | 26 | 32 | 35 | 10

Now, compare 32 and 35.

13 | 26 | 32 | 35 | 10

Here, 35 is greater than 32. So, there is no swapping required as they are already sorted.
Now, the comparison will be in between 35 and 10.

13 | 26 | 32 | 35 | 10

Here, 10 is smaller than 35 that are not sorted. So, swapping is required. Now, we reach at the end of the array. After first pass, the array will be -

13 | 26 | 32 | 10 | 35

Now, move to the second iteration.
Second Pass
The same process will be followed for second iteration.

13 | 26 | 32 | 10 | 35

13 | 26 | 32 | 10 | 35

13 | 26 | 32 | 10 | 35

Here, 10 is smaller than 32. So, swapping is required. After swapping, the array will be

13 | 26 | 10 | 32 | 35

13 | 26 | 10 | 32 | 35

Now, move to the third iteration.
Third Pass
The same process will be followed for third iteration.

13 | 26 | 10 | 32 | 35

13 | 26 | 10 | 32 | 35

Here, 10 is smaller than 26. So, swapping is required. After swapping, the array will be -

13 | 10 | 26 | 32 | 35

13 | 10 | 26 | 32 | 35

13 | 10 | 26 | 32 | 35

Now, move to the fourth iteration.
Fourth pass
Similarly, after the fourth iteration, the array will be -

10 | 13 | 26 | 32 | 35

Hence, there is no swapping required, so the array is completely sorted.

Q12.Write a program to implement bubble sort

```
#include<stdio.h>

Void main()

{

int a[50],i,j,n,temp;

printf("Enter size of Array");

scanf("%d",&n);

printf("Enter elements of array");

for(i=0;i<n;i++)

{
```

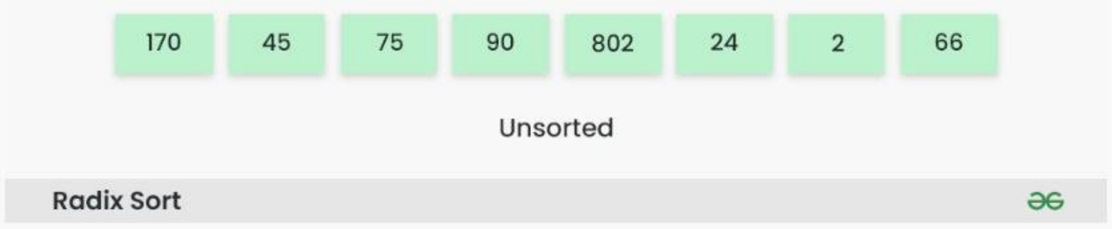
```
Scanf("%d",&a[i]);
}
for(i=0;i<n-1;i++)
{
for(j=0;j<n-i-1;j++)
{
if(a[j]>a[j+1])
{
Temp=a[j];
A[j]=a[j+1];
A{j+1}=temp;
}}}
Printf("sorted array");
For(i=0;i<n;i++)
{
Printf("%d",a[i];
}
}
```

Q13.DESCRIBE WORKING OF RADIX SORT. (4MARKS)

- In this article, we will discuss the Radix sort Algorithm. Radix sort is the linear sorting algorithm that is used for integers.
- In Radix sort, there is digit by digit sorting is performed that is started from the least significant digit to the most significant digit.
- The process of radix sort works similar to the sorting of students names, according to the alphabetical order. In this case, there are 26 radix formed due to the 26 alphabets in English.
- In the first pass, the names of students are grouped according to the ascending order of the first letter of their names
- After that, in the second pass, their names are grouped according to the ascending order of the second letter of their name. And the process continues until we find the sorted list. Now, let's see the algorithm of Radix sort.

Pranjal Save(Sy-comps)

To perform radix sort on the array [170, 45, 75, 90, 802, 24, 2, 66], we follow these steps:



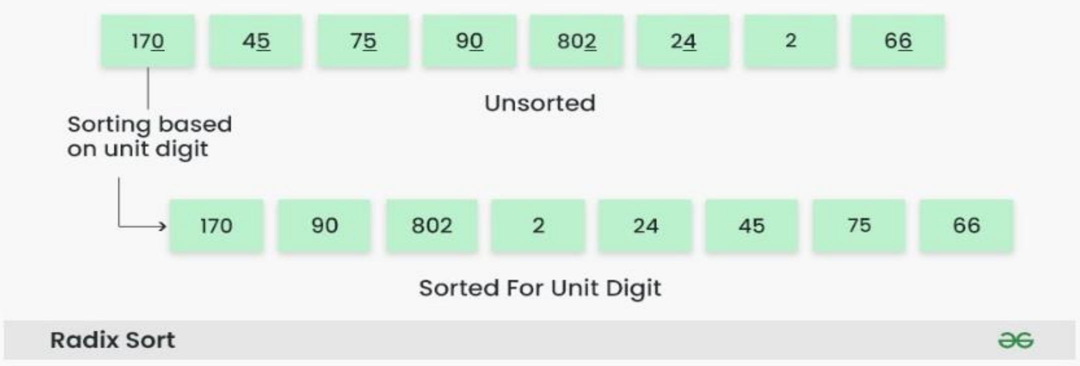
How does Radix Sort Algorithm work | Step 1

Step 1: *Find the largest element in the array, which is 802. It has three digits, so we will iterate three times, once for each significant place.*

Step 2: *Sort the elements based on the unit place digits (X=0). We use a stable sorting technique, such as counting sort, to sort the digits at each significant place.*

Sorting based on the unit place:

- *Perform counting sort on the array based on the unit place digits.*
- *The sorted array based on the unit place is [170, 90, 802, 2, 24, 45, 75, 66].*



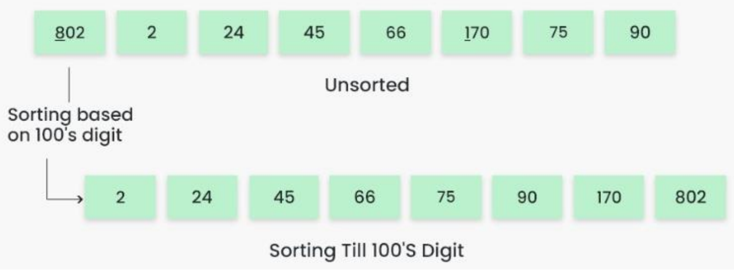
How does Radix Sort Algorithm work | Step 2

Pranjal Save(Sy-comps)

Step 4: *Sort the elements based on the hundreds place digits.*

Sorting based on the hundreds place:

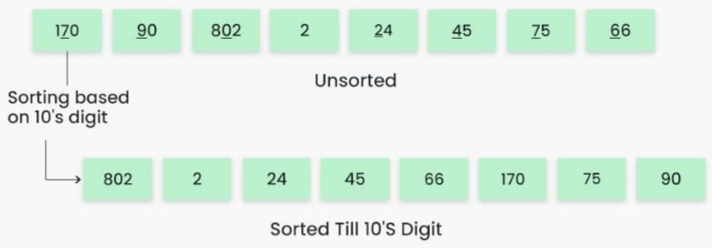
- *Perform counting sort on the array based on the hundreds place digits.*
- *The sorted array based on the hundreds place is [2, 24, 45, 66, 75, 90, 170, 802].*



Step 3: *Sort the elements based on the tens place digits.*

Sorting based on the tens place:

- *Perform counting sort on the array based on the tens place digits.*
- *The sorted array based on the tens place is [802, 2, 24, 45, 66, 170, 75, 90].*



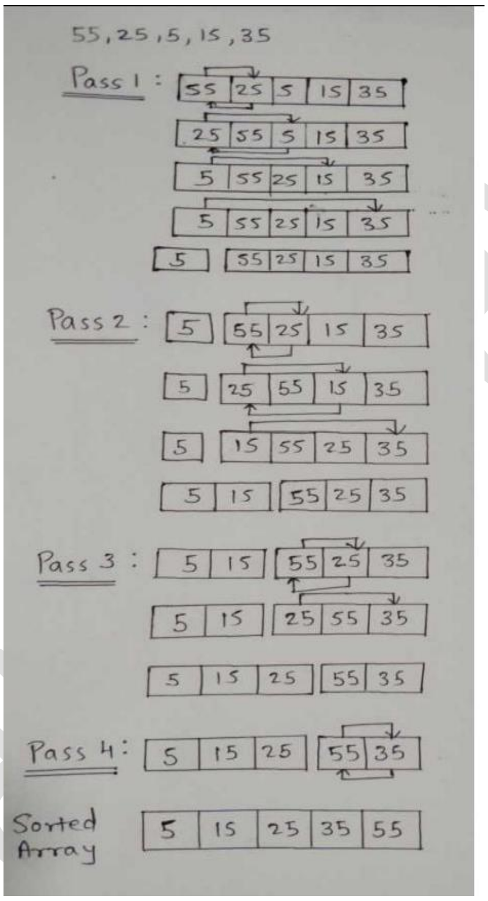
Step 5: *The array is now sorted in ascending order.*

The final sorted array using radix sort is [2, 24, 45, 66, 75, 90, 170, 802].

Pranjal Save(Sy-comps)

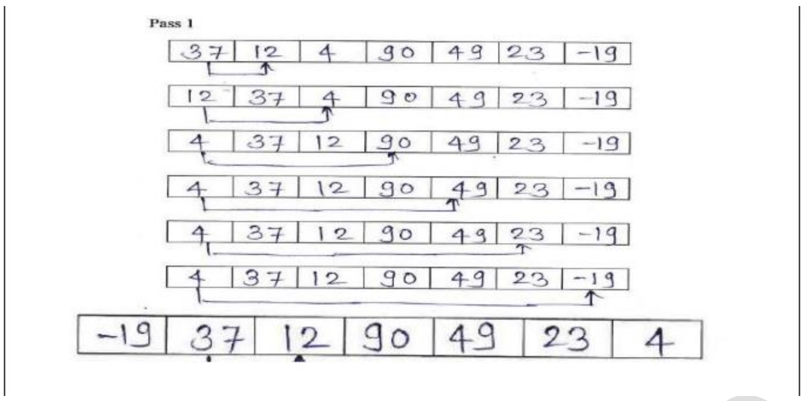
Q14. Describe working of selection sort method. Also sort given input list in ascending order using selection sort input list:- 55, 25, 5, 15, 35.(6marks)

Ans - Working of Selection sort: Selection Sort algorithm is used to arrange a list of elements in a particular order (Ascending or Descending). In selection sort, the first element in the list is selected and it is compared repeatedly with remaining all the elements in the list. If any element is smaller than the selected element (for ascending order), then both are swapped. Then we select the element at second position in the list and it is compared with remaining all elements in the list. If any element is smaller than the selected element, then both are swapped. This procedure is repeated till the entire list is sorted.

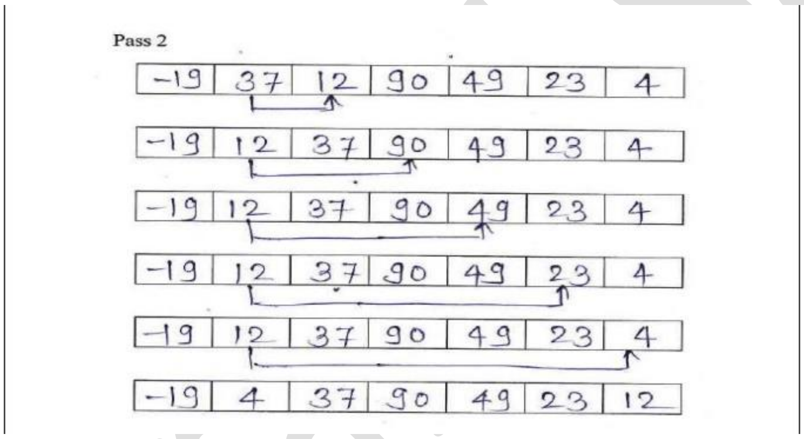


Pranjal Save(Sy-comps)

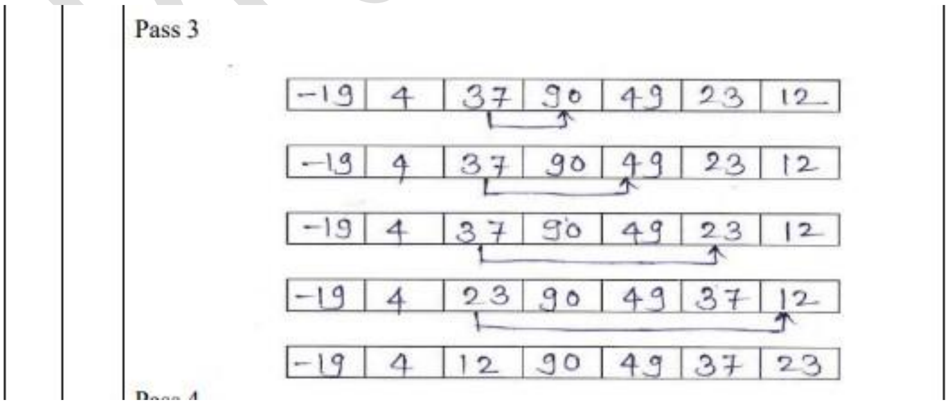
Q15. Elaborate the steps for performing selection sort for given elements of array. A={37,12,4,90,49,23,-19} (6marks)



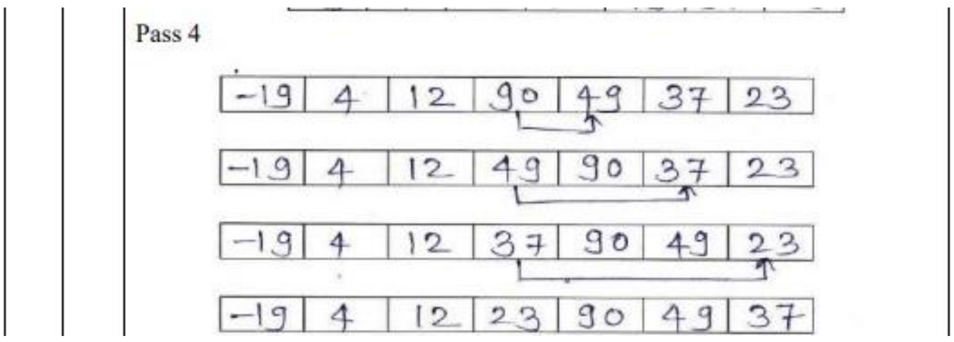
37	12	4	90	49	23	-19
12	37	4	90	49	23	-19
4	37	12	90	49	23	-19
4	37	12	90	49	23	-19
4	37	12	90	49	23	-19
4	37	12	90	49	23	-19
-19	37	12	90	49	23	4

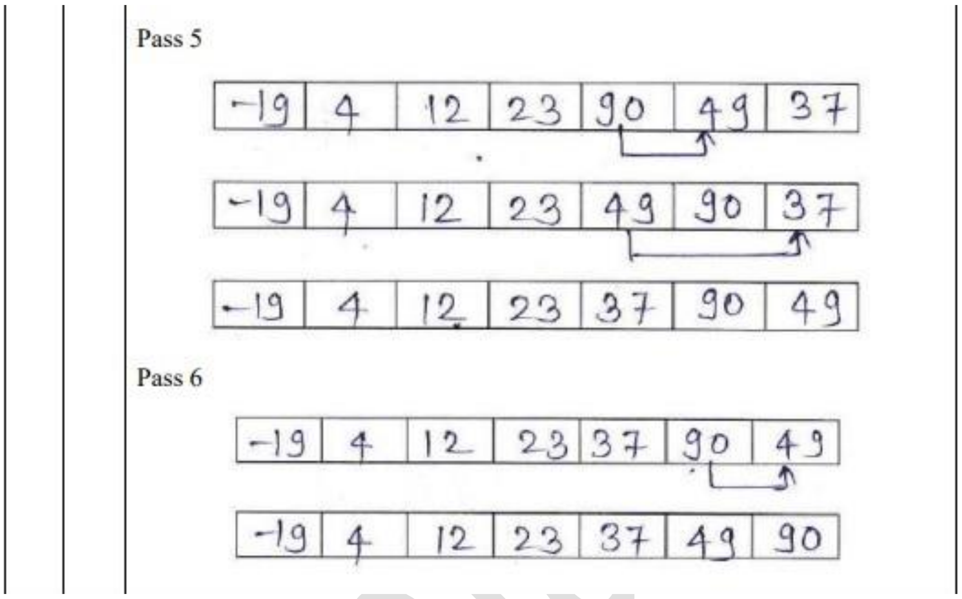


-19	37	12	90	49	23	4
-19	12	37	90	49	23	4
-19	12	37	90	49	23	4
-19	12	37	90	49	23	4
-19	12	37	90	49	23	4
-19	4	37	90	49	23	12



-19	4	37	90	49	23	12
-19	4	37	90	49	23	12
-19	4	37	90	49	23	12
-19	4	23	90	49	37	12
-19	4	12	90	49	37	23





Q16. Write a program for insertion sort.(4mark)

```
#include <stdio.h>
int main()
{
   int a[100], number, i, j, temp;

   printf("\n Please Enter the total Number of Elements  :  ");
   scanf("%d", &number);

   printf("\n Please Enter the Array Elements  :  ");
   for(i = 0; i < number; i++)
      scanf("%d", &a[i]);

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

```
printf("\n Insertion Sort Result : ");
   for(i = 0; i < number; i++)
   {
      printf(" %d \t", a[i]);
   }
   printf("\n");
   return 0;
}
```

Pranjal Save(Sy-comps)

Q17. Radix sort (4marks)

348,14,641,3851,74.