AIM:	Demonstrate the use of one-dimensional arrays to solve a given problem.
Program 1.A	

PROBLEM STATEMENT :	The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from the unsorted part and putting it at the beginning. The algorithm maintains two sub-arrays in a given array.
	1) The sub-array which is already sorted. 2) Remaining sub-array which is unsorted.
	Every iteration of selection sort, the minimum element (considering ascending order) from the unsorted sub-array is picked and moved to the sorted sub-array.
	Following example explains the above steps: arr[] = 64 25 12 22 11
	// Find the minimum element // and place it at beginning
	11 25 12 22 64
	// Find the minimum element // and place it at beginning of 11 12 25 22 64
	// Find the minimum element // and place it at beginning of 11 12 22 25 64
	// Find the minimum element // and place it at beginning of 11 12 22 25 64
	in arr[0...4]
	in arr[1...4] arr[1...4]
	in arr[2...4] arr[2...4]
	in arr[3...4] arr[3...4]

PROGRAM:	#include <stdio.h>
	
	void swap(int *xp, int *yp)
	{
	int temp = *xp;
	*xp = *yp;
	*yp = temp;
	}
	
	void selectionSort(int arr[], int n)
	{
	int i, j, min_idx;
	
	// One by one move boundary of unsorted subarray
	for (i = 0; i < n-1; i++)

```
        {
                // Find the minimum element in unsorted array
                min_idx = i;
                for (j = i+1; j < n; j++)
                if (arr[j] < arr[min_idx])
                        min_idx = j;

                // Swap the found minimum element with the first element
                swap(&arr[min_idx], &arr[i]);
        }
}
// Driver program to test above functions
int main()
{
   printf("Enter the number: ");
   int n;
   scanf("%d",&n);
   int arr[n];
   for (int i=0;i<n;i++)
      scanf("%d",&arr[i]);
   printf("\n");

        selectionSort(arr, n);
        printf("Sorted array: \n");
        for (int i=0;i<n;i++)
     printf("%d ",arr[i]);

  return 0;
}
```

RESULT:

```
Enter the number: 5
12 34 45 89 012

Sorted array:
12 12 34 45 89
```

Program 1.B

PROBLEM STATEMENT :	Perform search of a particular element on the above array using binary search. Binary Search will search a sorted array by repeatedly dividing the search interval in half. Begin with an interval covering the whole array. If

	the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half. Otherwise narrow it to the upper half. Repeatedly check until the value is found, or the interval is empty. We basically ignore half of the elements just after one comparison. 1. Compare x with the middle element. 2. If x matches with middle element, we return the mid index. 3. Else If x is greater than the mid element, then x can only lie in right half subarray after the mid element. So, we recur for right half. 4. Else (x is smaller) recur for the left half.

PROGRAM:	#include <stdio.h>
	void swap(int *xp, int *yp)
	{
	int temp = *xp;
	*xp = *yp;
	*yp = temp;
	}
	void selectionSort(int arr[], int n)
	{
	int i, j, min_idx;
	
	// One by one move boundary of unsorted subarray
	for (i = 0; i < n-1; i++)
	{
	// Find the minimum element in unsorted array
	min_idx = i;
	for (j = i+1; j < n; j++)
	if (arr[j] < arr[min_idx])
	min_idx = j;
	
	// Swap the found minimum element with the first element
	swap(&arr[min_idx], &arr[i]);
	}
	}
	int binary_search(int arr[],int n, int key){
	int low=0;
	int high=n-1;
	if (arr[low]==key)
	return low;
	else if (arr[high]==key)

```
    return high;
  while(low<=high){
    int midd=low+(high-low)/2;
    if (arr[midd]==key)
      return midd;
    else if(arr[midd]<key)
      high=midd-1;
    else
      low=midd+1;
  }
  return -1;
}
// Driver program to test above functions
int main()
{
  printf("Enter the number: ");
  int n;
  scanf("%d",&n);
  int arr[n];
  for (int i=0;i<n;i++)
    scanf("%d",&arr[i]);
  printf("\n");

   selectionSort(arr, n);
  //Taking key from user
  printf("Enter the number you want to search in array: ");
  int key;
  scanf("%d",&key);
  int index=binary_search(arr,n,key);
  if (index>=0)
    printf("Found");
  else
    printf("Not Found");
      return 0;
}
```

RESULT:

```
Enter the number: 4
12 34 56 78

Enter the number you want to search in array: 12
Found
```

Program 2	
PROBLEM STATEMENT:	insert and delete element from array. insert demo: sample i/p: [2 4 5 7 8 4 3 6 7] insert 3 at 4th position o/p: [2 4 5 3 7 8 4 3 6 7] delete demo : sample i/p: [2 4 5 7 8 4 3 6 7] delete 7th element o/p: [2 4 5 7 8 4 6 7] 3 gets deleted as it is 7th element

PROGRAM:	#include <stdio.h>
	//to Insert a element
	void inserstion(int arr[], int n,int num, int index){
	int arr2[n+1];
	for (int i=0;i<=index;i++)
	arr2[i]=arr[i];
	arr2[index-1]=num;
	for(int i=index;i<=n;i++)
	arr2[i]=arr[i-1];
	for (int i=0;i<n+1;i++)
	printf("%d ",arr2[i]);
	}
	//to delete a element
	void deletes(int arr[], int n,int pos){
	int arr2[n-1];
	for(int i=0;i<pos-1;i++)
	arr2[i]=arr[i];
	for(int i=pos;i<n;i++)
	arr2[i-1]=arr[i];
	for (int i=0;i<n-1;i++)
	printf("%d ",arr2[i]);
	}
	int main()
	{
	printf("Enter the number: ");
	int n;
	scanf("%d",&n);
	int arr[n];
	for (int i=0;i<n;i++)
	scanf("%d",&arr[i]);
	printf("\n");

```
    printf("Enter the value and the index of a number you want to insert: ");
    int num, index;
    scanf("%d %d",&num,&index);
    inserstion(arr,n,num,index);
    printf("\n================================");
    printf("\nThe array is:\n");
    for (int i=0;i<n;i++)
        printf("%d ",arr[i]);
    printf("\nEnter the element to delete:");
    int pos;
    scanf("%d",&pos);
    deletes(arr,n,pos);
    return 0;
}
```

RESULT:

```
Enter the number: 4
1 2 3 4

Enter the value and the index of a number you want to insert: 2 2
1 2 2 3 4
================================
The array is:
1 2 3 4
Enter the element to delete:3
1 2 4
```

CONCLUSION:	We learned to use one-dimensional arrays to solve a given problem