1)

```
#include<stdio.h>
void sort(int a[],int n)
{
        int i,j,temp;
        for(i=0;i<n;i++)
        {
                for(j=i+1;j<n;j++)
                {
                        if(a[i]<a[j])
                        {
                                temp=a[i];
                                a[i]=a[j];
                                a[j]=temp;
                        }
                }
        }
}
int binary(int a[],int e,int n)
{
        int i=0,j=n-1,mid;
        while(i<=j)
        {
                mid=(i+j)/2;
                if(a[mid]==e)
                        return mid+1;
                else
                {
                        if(e<a[mid])
                                j=mid-1;
                        else
                                i=mid+1;
                }
        }
        if(i>j)
        {
                return 0;
        }
}
int main()
{
        int n,i,a[20],f,e,m1,m2;
        printf("enter the no of elements of array");
        scanf("%d",&n);
        printf("enter the elements of array\n");
        for(i=0;i<n;i++)
```

```
            scanf("%d",&a[i]);
        sort(a,n);
        for(i=0;i<n;i++)
            printf("%d",a[i]);
        printf("enter the element to find in array");
        scanf("%d",&e);
        f=binary(a,e,n);
        if(f!=0)
        {
            printf("element is found at %d position",f);
        }
        else
        {
            printf("element not found\n");
        }
        printf("enter the position of array to find sum and product\n");
        scanf("%d%d",&m1,&m2);
        m1--;
        m2--;
        printf("the sum is %d",a[m1]+a[m2]);
        printf("the product is %d",a[m1]*a[m2]);
    }
```

2

```
    * C program for Merge Sort */
#include<stdlib.h>
#include<stdio.h>

// Merges two subarrays of arr[].
// First subarray is arr[l..m]
// Second subarray is arr[m+1..r]
void merge(int arr[], int l, int m, int r)
{
    int i, j, k;
    int n1 = m - l + 1;
    int n2 =  r - m;

    /* create temp arrays */
    int L[n1], R[n2];

    /* Copy data to temp arrays L[] and R[] */
    for (i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (j = 0; j < n2; j++)
```

```
        R[j] = arr[m + 1+ j];

    /* Merge the temp arrays back into arr[l..r]*/
    i = 0; // Initial index of first subarray
    j = 0; // Initial index of second subarray
    k = l; // Initial index of merged subarray
    while (i < n1 && j < n2)
    {
        if (L[i] <= R[j])
        {
            arr[k] = L[i];
            i++;
        }
        else
        {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    /* Copy the remaining elements of L[], if there
       are any */
    while (i < n1)
    {
        arr[k] = L[i];
        i++;
        k++;
    }

    /* Copy the remaining elements of R[], if there
       are any */
    while (j < n2)
    {
        arr[k] = R[j];
        j++;
        k++;
    }
}

/* l is for left index and r is right index of the
   sub-array of arr to be sorted */
void mergeSort(int arr[], int l, int r)
{
    if (l < r)
    {
        // Same as (l+r)/2, but avoids overflow for
        // large l and h
        int m = l+(r-l)/2;

        // Sort first and second halves
        mergeSort(arr, l, m);
        mergeSort(arr, m+1, r);

        merge(arr, l, m, r);
    }
}

/* UTILITY FUNCTIONS */
/* Function to print an array */
void printArray(int A[], int size)
{
    int i;
    for (i=0; i < size; i++)
        printf("%d ", A[i]);
    printf("\n");
}

/* Driver program to test above functions */
int main()
{
    int arr[5];
    int i;
    int arr_size = sizeof(arr)/sizeof(arr[0]);
    for(i=0;i<arr_size;i++){
       printf("enter the elements");
       scanf("%d",&arr[i]);
    }

    printf("Given array is \n");
    printArray(arr, arr_size);

    mergeSort(arr, 0, arr_size - 1);

    printf("\nSorted array is \n");
    printArray(arr, arr_size);
    int k;
    printf("enter the value of k");
    scanf("%d",&k);
    int fromfirst = arr[k-1];
    int fromlast = arr[5-(k)];
    printf("%d",fromlast*fromfirst);
```

```
   return 0;
}
```

3)
[11:26 am, 04/05/2020] Pavan Raghavendra: Selection Sort
The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning. The algorithm maintains two subarrays in a given array.

1) The subarray which is already sorted.
2) Remaining subarray which is unsorted.

In every iteration of selection sort, the minimum element (considering ascending order) from the unsorted subarray is picked and moved to the sorted subarray.
[11:26 am, 04/05/2020] Pavan Raghavendra: Following example explains the above steps:

```
arr[] = 64 25 12 22 11

// Find the minimum element in arr[0...4]
// and place it at beginning
11 25 12 22 64

// Find the minimum element in arr[1...4]
// and place it at beginning of arr[1...4]
11 12 25 22 64

// Find the minimum element in arr[2...4]
// and place it at beginning of arr[2...4]
11 12 22 25 64

// Find the minimum element in arr[3...4]
// and place it at beginning of arr[3...4]
11 12 22 25 64
```

[11:26 am, 04/05/2020] Pavan Raghavendra: Insertion Sort
Insertion sort is a simple sorting algorithm that works the way we sort playing cards in our hands.
Algorithm

```
// Sort an arr[] of size n
insertionSort(arr, n)
Loop from i = 1 to n-1.
……a) Pick element arr[i] and insert it into sorted sequence arr[0…i-1]
```

[11:26 am, 04/05/2020] Pavan Raghavendra: Example:
12, 11, 13, 5, 6

Let us loop for i = 1 (second element of the array) to 4 (last element of the array)

i = 1. Since 11 is smaller than 12, move 12 and insert 11 before 12
11, 12, 13, 5, 6

i = 2. 13 will remain at its position as all elements in A[0..I-1] are smaller than 13
11, 12, 13, 5, 6

i = 3. 5 will move to the beginning and all other elements from 11 to 13 will move one position ahead of their current position.
5, 11, 12, 13, 6

i = 4. 6 will move to position after 5, and elements from 11 to 13 will move one position ahead of their current position.
5, 6, 11, 12, 13

4)

```
  #include <stdio.h>
void main()
{
 int a[100],n,i,j,temp,sumo=0,prod=1,m;
 printf("Enter number of elements\n");
 scanf("%d",&n);
 printf("Enter %d integers\n", n);
 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("\nSorted list in ascending order:\n");

 for(i=0;i<n;i++)
 {
   printf("%d\n", a[i]);
 }
 printf("the alternate order is");
 for(i=0;i<n;i++)
 {
     if(i%2==0)
     {
         printf(" %d ",a[i]);
     }
 }
 for(i=0;i<n;i++)
 {
     if(i%2!=0)
     {
         sumo=sumo+a[i];
     }
 }
 printf("\nSum of odd Index is %d ",sumo);
 for(i=0;i<n;i++)
 {
     if(i%2==0)
     {
         prod=prod*a[i];
     }
 }
 printf("\nproduct of odd Index is %d ",prod);
 printf("\nEnter the value of m\n");
 scanf("%d",&m);
 for(i=0;i<n;i++)
 {
     if(a[i]%m==0)
     {
         printf("%d ",a[i]);
     }
 }

}
```

5)

```
    #include <stdio.h>
int recursiveBinarySearch(int array[], int start_index, int end_index, int element){
   if (end_index >= start_index){
      int middle = start_index + (end_index - start_index )/2;
      if (array[middle] == element)
         return middle;
      if (array[middle] > element)
         return recursiveBinarySearch(array, start_index, middle-1, element);
      return recursiveBinarySearch(array, middle+1, end_index, element);
   }
   return -1;
}
int main(void){
   int array[] = {1, 4, 7, 9, 16, 56, 70};
   int n = 7;
   int element = 9;
   int found_index = recursiveBinarySearch(array, 0, n-1, element);
   if(found_index == -1 ) {
      printf("Element not found in the array ");
   }
   else {
      printf("Element found at index : %d",found_index);
   }
   return 0;
}
```