1. Take the elements from the user and sort them in descending order and do the following.

a. Using Binary search find the element and the location in the array where the element is asked from user.

b. Ask the user to enter any two locations, print the sum and product of values at those location in the sorted array.

Sol:-

```
#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)
{
```

Teacher's Signature : _____

```
    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. Sort the array using merge sort where elements are taken from the user and find the product of K^{th} elements from first and last where K is taken from the user.

Sol:

```
#include<stdio.h>

Void  merge sort(int a[], int i, int j);
Void  merge (int a[], int i1, int j1, int i2, int j2);

int main ()
{
  int a[30], n, i, K, prod;
  printf ("\n Enter the number of elements :\n");
  Scanf ("%d", &n);
  printf ("\n Enter array elements :");
  for (i=0; i<n; i++)
  {
    Scanf ("%d", &a[i]);
  }
  Merge sort (a, 0, n-1);
  printf ("\n Sorted array is :");
  for (i=0; i<n; i++)
  {
    printf ("%d\t", a[i]);
  }
  return 0;
  printf ("\n Enter the value of K less than %d:", n);
```

Teacher's Signature :

```
    scanf("%d", &k);
    prod = a[k] * a[n-k];
    printf("\n Product of two elements is %d", prod);
}
void mergesort (int a[], int i, int j)
{
    int mid;
    if (i<j)
    {
        mid = (i+j)/2;
        mergesort(a, i, mid);
        mergesort (a, mid+1, j);
        merge (a, i, mid, mid+1, j);
    }
}
void merge (int a[], int i1, int j1, int i2, int j2)
{
    int temp[50];
    int i, j, k;
    i = i1;
    j = i2;
    k = 0;
    while (i<=j1 && j<=j2)
    {
        if (a[i]<a[j])
            temp[k++] = a[i++];
        else
            temp[k++] = a[j++];
    }
```

Teacher's Signature :

```
while (i<=j1)
  temp [k++] = a [i++];
while (j<=j2)
  temp [k++] = a [j++];
for (i=i1, j=0; i<=j2; i++, j++)
    a[i] = temp [j];
}
```

3. Discuss Insertion sort and selection sort with examples.

Sol:-

1. Insertion sort:- It 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]

Example:- 12, 11, 13, 5, 6
Let us loop for i = 1 (second element of array) to 4 (last element of array)

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

i = 2. Since 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

Teacher's Signature :

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.

2. 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.

=> 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.
// 25 12 22 64

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

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

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

4. Sort the array using bubble sort where elements are taken from the user and display the elements.
i. In alternate order.
ii. Sum of elements in odd positions and product of elements in even.
iii. Elements which are divisible by m where m is taken from the user.

Soln

```
#include <stdio.h>
Void main ()
{
 int a[100] , n , i, j, temp, sum0 = 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-1 ; j++ )
    {
       if (a[j] >a[j+1])
       {
         temp = a[j];
         a[j] = a[j+1];
```

```
        a[j+1] = temp;
      }
    }
  }
printf("\n Sorted 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)
   {
     sum0 = sum0 + a[i];
   }
 }
 printf("\n Sum of odd index is %d", sum0);
 for(i=0; i<n; i++)
 {
   if(i%2==0)
```

Expt. No. _____ Page No. _____

```
    {
      prod = prod * a[i];
    }
}
printf("\n Product of odd indese is %d", prod);
printf("\n Enter the value of m\n");
scanf("%d", &m);
for (i=0; i<n; i++)
{
  if (a[i] % m == 0)
  {
    printf("%d", a[i]);
  }
 }
}
```

5. Write a recursive program to implement binary search?

Sol:

```
#include <stdio.h>
int recursiveBinarySearch(int array[], int start_indexe,
                          int end_indexe, int element){
  if (end_indexe >= start_indexe){
    int middle = start_indexe + (end_indexe - start_indexe)/2;
    if (array[middle] == element)
      return middle;
    if (array[middle] > element)
      return recursiveBinarySearch(array, start_indexe, midde-1,
                                   element);
    return recursiveBinarySearch(array, middle+1, end_indexe,
                                 element);
  }
  return -1;
}
int main(Void){
  int array[] = {1, 4, 7, 9, 16, 56, 70};
  int n = 7;
  int element = 9;
  int found_indexe = recursiveBinarySearch(array, 0, n-1,
                                           element);
  If (found_indexe == -1){
    printf("Element not found in the array");
  }
  else {
```

```
        printf("element found at indese: %d", found_indese);
    }
    return 0;
}
```