ASSIGNMENT 6

MILIND GUPTA AP19110010498

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 locations in the sorted array.

⇒

```
#include <stdio.h>
int main()
{
int i, low, high, mid, n, key, arr[100], temp, j, one, two, sum, product;
printf(" Enter the number of elements in array");
scanf ("%d", &n);
printf ("Enter %d integers", n);
for (i=0 ; i<n ; i++)
scanf ("%d", &arr[i]);
for(i=0 ; i<n; i++)
{
  for (j=i+1; j <n ; j++)
  {
    if (arr[i] < arr[j])
    {
      temp= arr[i];
      arr[i ] = arr[j] ;
      arr[j] =tmp;
    }
  }
}
```

```
printf("\n Elements of array is sorted in descending way: \n");
for (i=0; i<n; i++)
{
  printf("%d", arr[i]);
}
printf("Enter value to find");
scanf("%d", &key);
low=0;
high= n-1;
mid= (low+high)/2;
while(low <=high) {
if (arr[mid] >key)
low= mid +1;
else if (arr[mid]==key){
printf("%d found at location %d", key, mid+1);
break;
}
else
high h= mid-1;
mid= (low+ high) /2;
}
if (low> high)
{
printf("Not found! %d isn't present in the list ", key);
}
printf("\n");
printf("Enter two locations to find sum and product ");
scanf("%d", &one);
scanf("%d", &two);
```

```
sum = (arr[one] + arr[two]);
product = (arr[one] * arr[two]);
printf ("The sum of elements = %d", sum);
printf ["The product of elements = %d", product);
return 0;
}
```

Output:-

```
Enter number of elements in array 4
 Enter 4 digits 3
 7
 1
 9
Element of array is sorted in descending order:
 9 1 7 3  Enter the value to find 1
 1 found at location 2
Enter two locations to find sum and product of
 the elements 1
  2
  The sum of elements = 10
The Product of elements = 9
```

2. Sort the elements using merge sort where elements are taken from the user and find the product of kth elements from first ~~first~~ and last where k is taken from the user.

⇒
```
#include<stdio.h>
#include<stdlib.h>
void merge(int arr[], int l, int m, int r)
{
 int i, j, k;
 int n1 = m-l+1;
 int n2 = r-m;
 int L[n1], R[n2];
 for (i=0; i<n1; i++)
    L[i] = arr[l+i];
    for (j=0; j<n2; j++)
    R[j] = arr[m+1+j];
 i=0;
 j=0;
 k=l;
 while (i<n1 && j<n2)
 {
    if (L[i] <= R[j])
    {
       arr[k] = L[i]
       i++;
    }
    else
    {
       arr[k] = R[j];
       j++
    }
    k++;
 }
}
```

```
while (i<n1)
{
 arr [k]= L[i]
  i++;
  k++;
 }
 while (j <n2)
 {
    arr[k] = R[j];
     j++;
     k++;
   }
}
 void merge_sort(int arr[], int l, int r)
  {  if (l<r)
     {
       int m= l+ (r-l) / 2;
       merge_sort (arr, l, m);
        mergesort( arr, m+1, r);
        merge (arr, l, m, r);
      }
   }
   void printArray(int A[], int size)
   {  int i;
      for (i=0 ; i< size ; i++)
         printf("%od ",A[i]);
      printf("\n");
   }
   int main()
   {  int arr[5];
      int i;
      int arr_size = sizeof(arr) / size_of(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 ("\n Sorted 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;
}
```

Output:- Enter 5 elements for sorting

9
7
6
3
1

Your data: 9 7 6 3 1

sorted data 1 3 6 7 9

Find the product of Kth elements from first and last where K

2

27

(3) Discuss Insertion sort and selection sort with examples:-

⇒ Insertion sort is a simple algorithm that builds the final sorted array (or list) one item at a time. It is less efficient on large lists than others. Insertion sort iterates, taking one input element each time and grows a sorted output list.

ex:- Take array:-

0	1	2	3	4
25	17	31	13	2

⇒ First iteration

0	1	2	3	4
25	17	31	13	2

⇓

0	1	2	3	4
17	25	31	13	2

second iteration remains same as 31 > 25, so swapping

⇓

17	25	31	13	2

Third iteration

13	17	25	31	2

4th iteration

2	13	17	25	31

Selection Sort :- It is an in-place comparision sorting algorithm. It has an $O(n^2)$ time complexity, which makes it inefficient on large list. It is noted for it's simplicity and has performance advantages.

Ex:-

Take an array:-

14	33	27	10	35	19

first element is 14 and smallest is 10, replacing 14 with 10.

10	33	27	14	35	19

↳ 10 appears at first position with min. value.

start scanning from second position 33 with lowest value, i.e- 14.

10	14	27	33	35	19

continuing the iteration

10	14	19	27	33	35

(4) Sort the array using bubble sort where elements are taken from the user and display the elements.

(i) alternate order.

(ii) Sum of elements in odd positions and products of elements in even position.

(iii) Elements which are divisible by m where m is taken from the user.

⇒
```
#include<stdio.h>
void main()
{
 int a[100],n,i,j,temp, sumo=0; prod=1, m;
 printf("Enter number of elements");
 scanf("%d", &n);
 printf("Enter %d integers", 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("\n sorted list in ascending order");
for(i=0; i<n ; i++)
{
 printf("%d" , 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("\n sum of odd Index is %d", sumo);
for(i=0; i<n ; i++)
{
   if (i%2 ==0)
   {
     prod = prod* a[i];
   }
}
```

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

Output:- Enter total number of element : 3
9
4
Sorting array using bubble sort
elements sorted successfully.
Sorted list in ascending order
3
4
8
array element in alternate
3
8
sum of odd index = 4
product of odd index = 4

(5.) Write a recursive program to implement binary search:-

```
#include<stdio.h>
int recursiveBinarySearch(int array[], int start_index, intend_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)
recursive
    return recursiveBinarysearch(array, start_index, middle-1, element);
 return recursiveBinary search (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 ");
   }
   else {
      printf ("Element found at index :%d", found_index);
   }
   return 0;
   }
```

Output :- Enter size of array: 4
Enter values in sorted :
1
2
3
4

Enter a value to be search 3
Element found at index : 2