ASSIGNMENT

G. Anuhya
CSE-F
AP19110010962

1) Take the element 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 where elements are taken from the user any two locations print the sum and product of values at those locations in the sorted array.

Code:-

```
#include <stdio.h>
 void sort(int a[], int n)
 {
     int i, j, temp;
     for (i=0; i<n; i++)
     {
         for(j=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 element 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 elements to find in array");
    scanf("%d", &e);
    f = binary (a, e, n);
    if(f != 0)
    {
        printf("elements is found at %d position", f);
    }
    else
    {
        printf("element not found");
    }
    printf("enter the position of array to find sum
                and product in");
    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>
// Merge 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 array */
    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[i];
   j++;
}
 k++;
}

 while (i<n1)
 {
  arr[k] = L[i];
  i++;
  k++;
 }
  while (i<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);
    merge sort (arr, m+1, r);
    merge (arr, l, m, r);
   }
 }
```

```
Void printArray(int A[], int size)
{
  int i;
  for(i=0; i<size; i++)
     Printf("%d ", A[i]);
   Printf("\n");
}

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("\n Sorted array is \n");
    Printf(Array(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) 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 begining. 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 (conserding ascending order) from the unsorted subbarray.

arr [] = 68 21 13 19 10

// find the minimum element in arr [0...4]
// and place it in the begining [0--4]

10 21 13 19 68

// find the minimum element in arr [1--4]
// and place it at begining [1--4]

10 13 21 19 68

// find the minimum element in arr [2...4]
// and place it at begining of arr [2--4]

10 13 19 21 64

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

10 13 19 21 64

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
insertion sort (arr, n)
loop from i = 1 to n-1

a) Pick element arr[i] and insert into sorted sequence arr[0 .. i-1]

Example: 13 10 15 7 8

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

i = 1 since 10 is smaller than 13, move 13 and insert 10 before 13

10 13 15 7 8

i = 2 since 13 will remain at its position as all elements in A[0 .. 1-1] are smaller than 13

10 13 15 7 8

i = 3 7 will move to the begining and all the other elements from 10 to 15 will move one position ahead of the current position.

i = 4 8 will move to the position after 7 and elements from 10 to 15 will move one position ahead from their current positions.

7 8 10 13 15

1)

```
#include <stdio.h>
void main()
{
 int a[100], n, i, j, temp, sum=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]);
   }
}
     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("\n product of odd index is %d", Prod);
   Printf("\n Enter the value of m");
   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 = recursive Binary Search
                        (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;
}
```