ASSIGNMENT- 6

NAME :- K. ANUGNA SAI
ROLL NO :- AP19110010414
CSE- F

(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 2 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= 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 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 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 Kth elements from first and last where K is taken from the user?

Code:-

```
#include <stdlib.h>
#include <stdio.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 mergeSort (int arr[], int l, int r)
{
  if(l<r)
{
```

```
    int m = l+(r-l)/2;
    mergeSort(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("%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");
  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) Discuss Insertion Sort and Selection Sort with Examples.

Code:-

Insertion Sort:-

Insertion Sort works by inserting the set of values in the existing sorted file. It constructs the sorted array by inserting a single element at a time. This process continues till whole array is sorted in same order. The primary concept behind insertion sort is each item into its appreciate place in the final list. The insertion sort method saves an effective amount of memory. The advantage of insertion sort is it works until there are elements in the unsorted set. Easily implemented and very efficient when used with small sets of data. It is faster than other sorting techniques. The best case complexity of insertion sort is O(n) times i.e. when the array is previously sorted.

For example:-

If we have the array as {40, 10, 50, 70, 30} and we apply insertion sort to sort the array, then the resultant array after each iteration will be as

Original array: {40, 10, 50, 70, 30}

Array after 1st iteration is: 10 → 40 → 50 → 70 → 30.

Array after 2nd iteration is :- 10 → 40 → 50 → 70 → 30.

Array after 3rd iteration is :- 10 → 40 → 50 → 70 → 30.

Array after 4th iteration is :- 10 → 30 → 40 → 50 → 70.

Selection Sort:-

Selection sort is another algorithm that is used for sorting. This sorting algorithm, iterates through the array and finds the smallest number in the and swaps it with it first element if it is smaller than the first element. Next, it goes on to the second element and so on until all elements are sorted.

Example of Selection Sort:-

Consider the array: [10,5,2,1].

The first element is 10. The next part we must find the smallest number from the remaining array. The smallest number from 5 2 and 1 is 1. Save replace 10 by 1.

The new array is [1,5,2,10] Again this process is repeated.

The run time complexity of selection sort in $O(n^2)$. Advantage of Selection-Sort is no additional storage is required beyond what is needed to hold the original list.

(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 in elements in even positions.

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

Code:-

```
#include<stdio.h>
void bubble sort(int ar[], int n)
{
    temp =ar[j];
    ar[j] = ar[j+1];
    ar[j+1] = temp;
}
}
```

```
int main()
{
 int size, i;
 printf("Enter size of required array:");
 scanf("%d", &size);
 int arr[size];
 for(i=0; i<size; i++)
 {
   printf("Enter Element:");
   scanf("%d", &arr[i]);
 }
 bubble sort(arr, size);
 printf("Sorted array:\n");
 for(i=0; i<size; i++)
 {
 printf("%d", arr[i]);
 printf("\t");
 }
 printf("\n**MENU**\n");
 printf("1.Display Elements in alternate order\n");
 printf("2. Sum of Elements in odd position and product of
        Elements in even positions\n");
 printf("3.Divisible by m\n");
 printf("Enter choice:");
 scanf("%d", &op);
 switch(op)
 {
  case 1:
 for(i=0; i<size; i+=2)
 {
   printf("%d\t", arr[i]);
 }
 case 2:
 for (i=0; i<size; i+=2)
 { sum= sum+arr[i];
 }
```

```
for (i=1; i<size; i+=2)
{
   product = product * arr[i];
}
printf("sum : %d\n", sum);
printf("product: %d\n", product);
```

Case 3:-

```
printf("Enter value m:");
scanf("%d", &m);
printf("Numbers divisible by %d are:\n", m);
for (i=0; i<size; i++)
{
 if (arr[i] % m == 0)
 {
  printf("%d\t", arr[i]);
 }
}
}
```

(5) Write a Recursive program to implement binary search?

Code:-

```
#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;
  }
```