ASSIGNMENT-6

M. Roop Sagar
AP19110010376
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 for 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 binary search (int arr [], int b, int x)
{
  if (b >= a) {
     int mid = a+(b-a)/2;
     if (arr[mid] == x)
        return mid;
     if (arr[mid] > x)
        return binary search (arr, a, mid-1, x);
     return binarysearch (arr, mid+1, b, x);
     }
     return -1;
  }
  int main()
  {
     int num;
     print f ("Enter thesize of array:");
     scanf ("%.d", & num);
     int i,j, a, val [Num], op, var, p1, p2, sum, pro;
     for (a=0; a <num ; a++)
     {
        print f ("Enter value :");
        scanf ("%. d', & val[a]);
```

```
}
for (i=0 ; i<num; ++i)
{
  for (j=i+1 ; j<num;++j)
  {
    if (val[i] < val[j])
    {
       a= val[i];
       val[i]= val[j]
       val[j]= a
    }
  }
printf("Array in descending order : ");
for (i=0 ; i<num; i++)
{
   printf("%d",val[i]);
}
printf ("\n** OPERATION _LIST **\n");
printf ("1. Find the value at entered position \n.
         2. Find the position of element \n.
         3. Printing sum & multiplication of values at
         entered positions");
printf ("\n Enter choice :\n");
scanf ("%d", &op);
switch (op)
{
    case 1:
    printf ("Enter the position to obtain value:");
    scanf ("%d", &var);
```

```
printf ("The value at %d position is %d", var, val[var]);
break;
case 2:
printf ("Enter element to find position:");
scanf ("%d", &var);
int result = binarySearch (val, 0, num-1, var);
(result == -1) ? printf ("Element is not present in array)
:printf ("Element is present at index %d", result);
return 0;
case 3:
printf ("\n Enter two positions to find sum and product
    of values \n");
scanf ("%d %d", &p1, &p2);
sum = val[p1] + val[p2];
pro = val[p1] * val[p2];
printf ("SUM = %d \n", sum);
print f ("MULTIPLICATION = %d", pro);
break;
}
}
```

8.) Sort the array using Merge sort where elements are taken from the user and find the product of kth element from first and last where k is taken from the user.

```
#include <stdlib.h>
# include <stdio.h>
void merge (int arr[], 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 tamp arrays back into array */
i=0 ; // Initial index of first i subarray
j=0 ; // initial index of second subarray
k = l ; // initial index of merged subaray
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 element of L[] if any */
while (i<n1)
{
  arr[k] = L[i];
  i++;
  k++;
}
/* copy the remaining elements of R[], if any */
while (j<n2)
{
  arr[k]= R[i];
  j++;
  k++;
}
}
void merge sort(int arr[], int l, int r)
{
  if(l<r)
  {
    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);
  }
}
/* Function to print an array */
void print Array(int A[], int size)
{
```

```
    int i;
    for (i=0 ; i<size ; i++)
      printf("%d ", A[i]);
    printf ("\n");
}
int main()
{
    int siz,v;
    printf ("Enter array size : ");
    scanf ("%d" ,&siz);
    int val[siz];
    for (v=0 ;v<siz; v++)
    {
      printf ("Enter value :");
      scanf ("%d", &val[v]);
    }
    printf ("Given array is \n");
    printArray (val, siz);
    mergesort (val ,0,siz-1)
    printf("\nSorted array is \n");
    printArray (val, siz);
    int k, f, l, p1 , p2 , temp;
    printf ("Enter the value of k to find the product of elements
     frint and last:");
    scanf ("%d", &k);
    p1 = p2 =1 ;
    for (f=0, f<=k, f++)
    {
      temp = val[f];
      p1 * = temp;
```

```
}
for (i = siz - 1; i >= k; i--)
{
  temp = val[i];
  p2* = temp
}
printf("Product of kth element from first and last are:
        %d %d", p1 p2);
}
```

3) Discuss insertion sort and selection sort with examples

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 in 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 work 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 first interation is : 10 → 40 → 50 → 70 → 30

Array after second interation is : 10 → 40 → 50 → 70 → 30

Array after third interation is : 10 → 40 → 50 → 70 → 30

Array after fourth interation 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 array 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.

examples 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 52 and 1 is 1. so, we replace 10 by 1. The new array is [1,5,2,10] Again this process is repeated

The run time complexity of selection sort is $O(n^2)$.

Advantages 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 of elements in even position (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[i] = ar[i+1];
  ar[j+1] = temp;
 }
}
 int main ()
 {
 int siz, i;
 printf ("Enter size of required array :");
 scanf ("%d", & siz);
 int arr[siz]
 for (i=0; i<siz ; i++)
 {
   printf ("Enter Element :")
   scanf ("%d", & arr[i]);
  }
  bubble sort (arr, siz);
  printf ("sorted array : \n");
  for (i=0 ; i<siz; i++)
  {
  printf ("%d", arr[i]);
```

```
printf("\t")
}
printf("\n** MENU **\n")
printf ("1. Display Element in alternate order \n");
printf ("2. Sum of element 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<Siz ; i+=2)
{
 case 1
for (i=0; i<Siz ; i+=2)
{
  printf ("%d\t", arr[i]);
}
case 2:
 for (i=0 ; i<siz ; i+=2)
 {
  sum= sum+arr[i];
 }
  for (i = 1; i < siz; i+=2)
  {
   product= product* arr[i];
  }.
  Print f("Sum : %d \n" sum);
```

```
        printf("product : %d \n",product);
case 3:
 printf("Enter value m:");
  scanf("%d", &m);
 printf("Numbers divisibles by %d are : \n",m);
 for(i=0 ;i<siz; i++)
 {
  if(arr[i] %m==0)
   {
     printf("%d\t",arr[i]);
    }
   }
  }
}
```

5) Write a recursive program to implement binary search?

```
# include <stdio.h>
int binary search (int a[] ,int low, int high, int x){
int mid = (low+high)/2;
if (low > high) return -1;
if (a[mid]==x) return mid;
 if (a[mid]<x)
  return binary search (a, low,mid+1 ,x);
Else
   return binary search (a, low ,mid-1 ,x);
  }
int main (void) {
 int a[100]; ien, pos, search_item;
```

```
printf("%d", &len);
printf("Enter the array element \n");
for (int i=0; i<len; i++)
scanf("%d", a[i]);
printf("Enter the element to search \n");
scanf("%d", &search_item);
pos = binarysearch(a, 0, len-1, search_item);
if(pos<0)
printf("cannot find the element %d in the array.
        \n", search_item);
else.
printf("position of %d in array is %d.\n", search_item, pos+1);
return 0;
}
```