J.Naveen
AP19110010378
CSE-H

DSA - Assignment-6

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

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

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

Program:

```
#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 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]);
printf (" the product

}
```

Scanned with CamScanner

output - 1 (a):

```
Enter the no. of elements of array : 6
Enter the elements of array:
8
4
6
7
2
5
Alternate order: 8 7 6 5 4 2
Enter the element to find the element in Array: 8
Element found at position 1.
```

output (b):

```
Enter the position of array to find sum and product
2
3
The sum is 10 and product is 24.
```

③ sort the array using merge sort where elements are taken from user and find the product of kth elements from first and last where k is from user.

Program:

```
#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);
merge sort (arr, m+1, r);
merge (arr, l, m, r);
}
}
void print array (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 = size of (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 ");
Print Array (arr , arr_size);
merge sort (arr, 0, arr_ size - 1);
Printf ("\n sorted array is \n");
Print Array (arr, arr_size);
int k ;
Print f (" Enter the value of k");
scanf ("%d', & k);
int from first = arr [k-1];
int from Last = arr [5-(k)];
printf ("%d", from last * from first);
return 0;
}
```

output:

```
enter the elements
                6
                4
                7
                2
                8
Given     array     is
          6 4 7 2 8

Sorted       array  is
             2 4 6 7 8
Enter the value of K 8
   Product is 0
```

③ Insertion sort :

Insert sort is live sorting technique which can deal with immediate data. This is an in-place comparison based sorting algorithm.

Example: the lower part of an array is maintained to be sorted. An element which is to be inserted in this sorted sub-list, has to find its appropriate place and then it has to be inserted there. Hence the name, insertion sort, the position to which it belongs in a sorted array.

It has O(n) best case complexity, O(n)^2 worst case performance, and O(n^2) Average performance.

Algorithm :

1. First step involves the comparison of the element in question with its adjacent element.
2. And if at every comparison reveals that the element can be inserted at a particular position, then space is created for it by shifting the other elements one position to the right and inserting the element at the suitable position.
3. the above procedure is repeated untill all the element in the array is at their apt position.

consider the following array: 25, 17, 31, 13, 2

First iteration : 17, 25, 31, 13, 2

Second iteration: 17, 25, 31, 13, 2

Third iteration: 13, 17, 25, 31, 2

Fourth iteration: 2, 13, 17, 25, 31

This is the final array after all the corresponding iterations and swapping of elements.

Selection sort:

Selection sort is a simple sorting algorithm. This selection sort sorts an array by repeatedly finding the minimum element. this sorting is an in-place comparison-based algorithm in which the list is divided into two parts, the sorted part at the left end and the unsorted part at the right end. Initially, the sorted part is empty and the unsorted part is the entire list.

It's main advantage is that it performs well on a small list. Furthermore, because it is an in-place sorting algorithm, no additional temporary storage is required beyond what is needed to hold the original list.

Algorithm:

1. set the first element as minimum.
   select first element as minimum.
2. Compare minimum with the second element.
3. After each iteration, minimum is placed in front of the unsorted list.
4. For each iteration, indexing starts from first unsorted element

```
(4) #include <stdio.h>
   void main()
   {
   int a[100] ,n, i , j, temp , sum 0=0, prod=1 , m;
   printf ("Enter the 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 ("\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)
{
Prod = prod * a[i] ;
}
}
Printf("\n product of odd Index 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]);
 }
 }
 }
```

Output:-

```
Enter the number of elements 5
Enter 5 integers
2
5
6
3
8
Sorted list in ascending order
2
3
5
6
8
The alternate order is 2 5 8 3 6
Sum of odd positions is 15
product of even positions is 18
Enter the value of m 2
Elements divisible by m 2 6 8
```

⑤ write a recursive program to implement binary search?

Program:-

```
#include <stdio.h>
void binary-search(int [], int ,int, int);
void bubble_sort (int [], int);
int main ()
{
 int key, size ,i;
 int list [25];
printf("Enter size of a list:");
scanf ("%d", &size);
printf("Enter elements \n");
for (i=0; i<size; i++).
{
 scanf ("%d", & list[i]);
}
bubble-sort(list, size);
printf("\n");
printf("Enter key to search \n");
scanf("%d", & key);
binary-search(list, 0, size , key);
}

void bubble-sort (int list [] , int size)
{
int temp, i, j;
for (i=0; i<size; i++)
{
 for (j=1; j< size; j++)
 {
  if (list[i] > list [j])
  {
        temp = list[i];
        list[i] = list[j];
        list[j] = temp;
      }
    }
  }
}
 void binary-search (int list[], int low, int high, int key)
 {
  int mid;
  if (low > high)
   {
    printf("key not found\n");
    return;
   }
   mid = (low + high) /2;
   if (list [mid] == key)
   {
    printf ("key found\n");
   } else if (list [mid] > key)
   {
    binary-search (list, low, mid-1, key);
   }
   else if (list [mid] < key)
   {
    binary-search (list, mid+1, high, key);
    }
 }
```

Output:

```
Enter size of a list : 5
Enter elements
4
7
8
9
6
Enter key to search 2
key not found.
```