Assignment - 6

T. Venkata Nagalakshmi
AP19110010223
CSE-G

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>
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 ele, int n)
{
   int i=0, j = n-1, mid;
   while (i<=j)
   {
      mid = (i+j)/2;
      if (a[mid] == ele)
            return mid+1;
      else
```

```
            {
                if (ele < a[mid])
                    j = mid-1;
                else
                    i = mid+1;
            }
        }
        if (i>j)
        {
            return 0;
        }
    }
int main()
{
    int n, i, a[50], b, ele, n1, n2;
    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", &ele);
    b= binary(a,ele, n);
    if (b!=0)
    {
        printf("element is found at %d position:", b);
    }
    else
    {
        printf("element not found: \n");
    }
Printf("enter position of array to find sum and product: \n");
```

(2)

```
scanf("%d %d", &n1, &n2);
n1--;
n2--;
printf("the sum is %d", a[n1] + a[n2]);
printf("the product is %d", a[n1] * a[n2]);
}
```

Output:

enter the no. of elements of array: 6

enter the elements of array: 33 34 23 45 24 50

50 45 34 33 24 23 enter the element to find in array: 5

element not found

enter the position of array to find sum and product: 5 3

the sum is 58 the product is 816.

(a) 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

```
# include <stdlib.h>
# include <stdio.h>
// merges two subarrays of arr[].
// first subarray is arr[0..p]
// second subarray is arr[p+1...q]
void merge (int arr[], int 0, int p, int q)
{
int i, j, k;
int n1 = p-0+1;
int n2 = q-p;
/* create temp arrays */
int 0[n1], a[n2];
for (i=0; i<n1; i++)
0[i] = arr(0+i];
for (j=0; j<n2; j++)
a[j] = arr[q+1+j];
i=0; // initial index of first subarray.
```

```
    j=0; // initial index of second subarray
    k=0; // initial index of merged subarray
    while (i<n1 && j<n2)
    {
    if (O[i] <= Q[j])
    {
    arr[k] = O[i];
     i++;
    }
    else
     {
      arr[k] = Q[j];
     j++;
    }
    k++;
    }
   while (i<n1)
    {
   arr[k] = O[i];
    i++;
   k++;
   }
  while (j<n2)
    {
     arr[k] = Q[j];
     j++;
     k++;
   }
  }
/* O is for left index and Q is right index of the
      sub array of arr to be sorted */
void merge sort (int arr[], int o, int q)
 {
 if (o<q)
```

```
    {
     // Same as (0+q)/2, but avoids overflow for
     // large o and h
     int p = o+(q-o)/2;
     // sort first and second halves
     merge sort (arr, o, p);
     Merge sort (arr, p+1, q);
     merge (arr, o, p, q);
     }
     }
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 = 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);
mergesort(arr, 0, arrsize -1);
printf("\n sorted array is : \n");
Print 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) Discuss insertion sort and selection sort with examples.

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
insertionsort (arr, n)
loop from i=1 to n-1.
a) pick element arr[i] and insert it into sorted sequence arr[0....i-1]

Ex:- 15, 10, 16, 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 15, move 15 and insert 10 before 15

10, 15, 16, 7, 8

i=2. 16 will remain at its position as all elements in A[0...i-1] are smaller than 16.

10, 15, 16, 7, 8

i=3. 7 will move to the begining and all other elements from 10 to 16 will move one position ahead of their current position.

7, 10, 15, 16, 8.

i=4. 8 will move to position after 7, and elements from 10 to 16 will move one position ahead of their current position.

7, 8, 10, 15, 16.

(4)

Selection Sort:

The selection sort algorithm sorts an array by repeatedly finding the minimum element 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 from the unsorted subarray is picked and moved to the sorted subarray.

Ex:- arr[] = 65, 26, 13, 23, 12

// find the minimum element in arr[0..4]
// and place it at begining
12, 26, 13, 23, 65

// find the minimum element in arr[1..4]
// and place it at begining of arr[1..4]
12, 13, 26, 23, 65

// find the minimum element in arr[2..4]
// and place it at begining of arr[2..4]
12, 13, 23, 26, 65

// find the minimum element in arr[3..4]
// and place it at begining of arr[3..4]
12, 13, 23, 26, 65.

(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 positions.
iii. elements which are divisible by m where m is taken from the user.

```
# include <stdio.h>
Void main ()
```

```
{
int a[50], n, i, j, temp, sum=0, prod=1, b;
printf(" enter no. 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(" 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)
```

```
        {
          sumo = sumo + a[i];
        }
      }
    printf(" 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 b \n");
  scanf(" %d ", &b);
  for (i=0; i<n; i++)
    {
      if (a[i] %m == 0)
        {
          printf(" %d ", a[i]);
        }
    }
}
```

output

```
enter no. of elements : 5
enter 5 integers: 45 56 23 40 78
Sorted list in ascending order :
23 40 45 56 78
the alternate order is 23 45 78
sum of odd index is 96
product of odd index is 2240
enter the value of b : 5
45.
```

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

```
# include < stdio.h >
int binarySearch (int A[], int low, int high, int x)
{
  if (low > high)
    return -1;
  int mid = (low + high)/2 ;
  if (x == A[mid])
      return mid;
  else if (x < A[mid])
      return binarySearch( A, low, mid-1, x);
  else
```

```
        return binarysearch (A, mid+1, high, x);
    }
// Recursive implementation of binary search
int main (void)
{
    int A[] = {3, 6, 7, 9, 10, 11};
    int target = 6;
    int n = size of (A) / size of (A[0]);
    int low = 0, high = n-1;
    int index = binary search (A, low, high, target);
    if (index != -1)
        printf(" element found at index %d", index);
    else
        printf(" element not found in the array");
    return 0;
}
```

Output:

element found at index 1.