Assignment 6
Searching and Sorting

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. Asle 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 b, int n)
{
    int i=0, j=n-1, mid;
    while(i<=j) {
        mid=(i+j)/2;
        if(a[mid]==b)
            return mid+1;
        else{
            if(b<a[mid])
                j=mid-1;
            else
                i=mid+1;
        }
    }
    if(i>j) {
        return 0;
    }
}
int main() {
    int n,i, a[20], c, b, s1, s2;
    printf("Enter the number of elements of array");
    scanf("%d", &n);
    printf("Enter the elements of array");
```

```
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",&b);
C=binary(a,b,n);
if(C!=0){
    printf("Element is found at position
                              %d", C);
}
else{
    printf("Element not found\n");
}
printf("Enter the position of array to
      find sum and product\n");
Scanf("%d%d",&S1,&S2);
 S1--;
 S2--;
printf("The Sum is %d",a[S1]+a[S2]);
printf("The Product is %d",a[S1]*a[S2]);
}
```

2. Sort the array using Merge sort where elements are taken from user and find the product of kth elements from the first and last where K is taken from the user.

```
#include <stdlib.h>
#include<stdio.h>
Void merge(int arr[], int l, int c, int e)
{
int i, j, k;
int n1 = c-l+1;
int n2 = e-c;
int L[n1], R[n2];
for (i=0; i<n1; i++)
L[i] = arr[i+1];
for (j=0; j<n2; j++)
R[j] = arr[c+1+j];

i=0;
j=0;
k=1;
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++;
}
}

Void mergeSort(int arr[], int l, int e){
if (r > l) {
int m = l + (e - l) / 2;
```

```
mergeSort(arr,l,c);
mergeSort(arr,c+1,e);
merge(arr,l,c,e);
}
}
Void printArray(int A[],intsize){
int i;
for(i=0;i<size;i++)
printf("%d",A[i]);
printf("\n");
}
int main()
{
int arr[7];
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[7-(k)];
printf("%d", fromlast * fromfirst);
return 0;
}
```

Output

```
Enter the elements: 65
Enter the elements: 98
Enter the elements: 32
Enter the elements: 25
Enter the elements: 15
Enter the elements: 46
Enter the elements: 74
Given array is
65 98 32 25 15 46 74
Sorted:
```

Sorted array is

15 25 32 46 65 74 98

Enter the value of K: 5

2080

3. Discuss Insertion sort and Selection sort with examples.

Insertion sort is a simple sorting algorithm that builds the final sorted array one item at a time. It is much less efficient on large lists than more advanced algorithms such as heapsort, merge sort.

The time complexity for insertion sort for worst case and best case are $O(n^2)$ and $O(n)$ respectively.

Algorithm

Sort of arr[] of size n insertionSort(arr, n)
loop from i = 1 to n-1.
Pick the element arr[i] and insert it into sorted sequence arr[0...i-1].

Example:-

	5	6	9	2	1
i =	0	1	2	3	4

let us loop for i = 1 to i = 4.

~~Since~~ i = 1, since 5 < 6, so 5 will as it is.

5	6	9	2	1

i = 2, since 9 > 6, so 6 will remain in its position.

5	6	9	2	1

i=3, 2 will move to the beginning and other elements 5 to 9 will move one position ahead of their current position.

i.e. | 2 | 5 | 6 | 9 | 1 |

i=4, 1 will move to the beginning and other elements from 2 to 9 will move one position ahead of their current position.

i.e. | 1 | 2 | 5 | 6 | 9 |
0 1 2 3 4

Selection sort is the sorting algorithm of an arry to finding the minimum element repeatedly from unsorted path and putting in the ~~sorted~~ beginning.

The time complexity for selection sort for worthand best case is $O(n^2)$ and $O(n^2)$ respectively.

Example:- | 13 | 22 | 11 | 5 | 9 |
0 1 2 3 4

Find the minimum element in arr[0...4] and place it at beggginning and displace the 5 position of previous element to the new element.

i.e. ~~5 22~~ | 5 | 22 | 11 | 13 | 9 |

Find the minimum element in arr[1..4] and place it at beginning of arr[1..4].

5	9	11	13	22

Find the minimum element in arr[2..4] and place it at beginning of arr[2..4].

5	9	11	13	22

Now all the elements has settled down in descending order. Repeat the same process for further rearrangements.

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[100],n,i,j,temp,sum=0,prod=1,m;
printf("Enter the elements");
Scanf("%d",&n);
printf("Enter integers %d\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");

for(i=0;i<n;i++){
  printf("%d\n",a[i]);
}
```

i. printf ("The alternate order is");
for(i=0; i<n; i++){
if (i%2==0){
printf("%d", a[i]);
}
}

ii. for (i=0; i<n; i++){
if (i%2!=0) {
Sum= Sum+a[i];
}
}
printf ("Sum of odd index is %d", Sum);
for (i=0; i<n; i++){
if (i%2==0) {
prod= prod*a[i];
}
}
printf("Product of ~~odd~~ even position is %d", prod);

iii. printf("Enter the value of m");
Scanf("%d", &m);
for (i=0; i<n; i++){
if(a[i]%m==0){
printf("%d", a[i]);
}}

5. Write a recursive program to implement binary search.

```
#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[] = {3, 13, 17, 5, 23, 57, 73};
```

```
int n=7 ;
int element= 23 ;
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;
}
```

Output

Element found at index : 4