CSE-H
R. Vamsi Krishna
AP19110010500

~~Assign~~
Assignment-6
(Searching and Sorting)

1)

A)

```
#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)
```

Scanned with CamScanner

```
{
    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 number of elements of array");
    Scanf("%d",&n);
    Printf("enter the elements of array\n");
    for(i=0;i<n;i++)
```

Scanned with CamScanner

```
        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]
}
```

Scanned with CamScanner

8)
```
#include <stdio.h>
#include <conio.h>
#define MAX_SIZE 5
void merge_Sort(int, int)
void merge_array(int, int, int, int);
int arr_Sort[MAX_SIZE];
int main() {
    int i, k, Pro=1;
Printf("Simple merge Sort Example functions
            and Array \n");
    Printf("\n Enter %d Elements for Sorting \n",
            MAX_SIZE);
  for(i=0; i<MAX_SIZE; i++)
  Scanf("%d", &arr_Sort[i]);
  Printf("\n Your Data :");
  for(i=0; i<MAX_SIZE; i++) {
      Printf("\t %d", arr_Sort[i]);
  }
 merge_Sort(0, MAX_SIZE-1);
Printf("\n\n Sorted Data:");
for(i=0; i<MAX_SIZE; i++) {
    Printf("\t %d", arr_Sort[i]);
}
```

Scanned with CamScanner

```
Printf("find the Product of Kth elements from first
        and last where K\n");
Scanf("%d",&K);
Pro = arr_Sort[K] * arr_Sort[MAX_SIZE-K-1];
Printf("Product = %d", Pro);
getch();
}
  Void merge_Sort(int i, int j){
  int m;
  if(i<j){
    m=(i+j)/2;
    merge_Sort(i,m);
     merge_Sort(m+1,j);
// Merging two arrays
merge_array(i,m,m+1,j);
}
}
Void merge_array(int a, int b, int c, int d){
 int t[50];
  int i=a, j=c, k=0;
  while(i<=b && j<=d){
  if(arr_Sort[i]<arr_Sort[j])
    t[k++]=arr_Sort[i++];
```

Scanned with CamScanner

```
else
 t[k++] = arr_sort[j++];
 }
// collect remaining elements
 while(i<=b)
  t[k++] = arr_sort[i++];
 while(j<=d)
  t[k++] = arr_sort[j++];
 for(i=a, j=0; i<=d; i++, j++)
  arr_sort[i] = t[j];
}
```

Scanned with CamScanner

3) Selection Sort

The Selection Sort algorithm Sorts an array by repeatedly finding the minimum element from unSorted Part and Putting it at the beginning. 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.

Example:

arr[] = 64 25 12 22 11

// Find the minimum element in arr[0---4]
// and Place it at beginning
11 25 12 22 64

// Find the minimum element in arr[1---4]
// and Place it at beginning of arr[1---4]
11 12 25 22 64

Scanned with CamScanner

// Find the minimum element in arr[2---4]
// and Place it at beginning of arr[2---4]
11 12 22 25 64
// Find the minimum element in arr[3---4]
// and Place it at beginning of arr[3---4]
11 12 22 25 64

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]

Example :

12, 11, 13, 5, 6

let us loop for i=1 (Second element of the array) to N (last element of the array)

Scanned with CamScanner

i=1. Since 11 is Smaller than 12, move 12 and insert 11 before 12

11, 12, 13, 5, 6

i=2. 13 will remain at its Position as all elements in A[0..i-1] are Smaller than 13

11, 12, 13, 5, 6

i=3. 5 will move to the beginning and all other elements from 11 to 13 will move one Position ahead of Current Position

5, 11, 12, 13, 6

i=4. 6 will move to Position after 5, and elements from 11 to 13 will move one Position ahead of their Current Position

5, 6, 11, 12, 13.

Scanned with CamScanner

```
#include <stdio.h>
void main()
{int a[100], n, i, j, temP, Sum=0, Prod=1, m;
Printf("Enter number of elements \n");
Scanf("%d", &n);
Printf("Enter %d integers \n", n);
for(i=0; i<n; j++)
{ Scanf("%d", &a[i]);
}
for(i=0; i<n-1; j++)
{ 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");
```

Scanned with CamScanner

```
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)
  {
    Sum0 = Sum0 + a[i];
  }
}
Printf("\n Sum of odd index is %d", Sum0)
for(i=0; i<n; i++)
{
  if(i%2==0)
{
```

Scanned with CamScanner

```
        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]);
    }
}
}
```

Scanned with CamScanner

5)
```
#include <stdio.h>
int recursive BinarySearch(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 recursive BinarySearch(array, start_index,
                                middle-1, element);
return recursive BinarySearch(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 = recursive BinarySearch(array,
                                0, n-1, element);
```

Scanned with CamScanner

```
if(found_index==-1){
  Printf("Element not found in the array");
}else{
  Printf("element found at index:%d",found_index);
}
return 0;
}
```

Scanned with CamScanner