Assignment - 6

Yogitha. G
AP19110010320

1)
```
#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 nos 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 for find in array");
  Scanf("%d", &e);
   f= binary(a,e,n);
    if(f!=0)
  {
    Printf("~~enter~~ element is found at %d Position", f);
  }
else
  {
    Printf("element not found\n");
  }
```

```
Printf("enter the position of array for 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]);
}
```

2)

```
#include<stdio.h>
#include <stdlib.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+j];
   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 (l < n1)
 {
     arr[k] = L[i]
     i++;
     k++;
 }
   while (j < n2)
  {
     arr[k] = R[j]
     j++;
      k++;
    }
 }
  void merge sort (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 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");
   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 from Last = arr[5-(k)];
 Printf("%d", from last* from first);
 return 0;
}
```

3) Selection sort :

The selection sort algorithm sorts an array by repeatedly finding the minimum element for 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 for the sorted subarray.

Examples:

arr[] = 64 25 12 22 11

find the minimum element in arr[0...4]
and Place it at beginning

11 25 12 22 64

arr [1...4] = 11 12 25 22 64

arr [2...4] = 11 12 22 25 64

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

```
insertion sort (arr, n)
loop from i = 1 to n - 1
```

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

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---1-1] 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 on Position ahead of their 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

4)

```
#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; i++)
 {
      Scanf("%d", &a[i]);
 }
   for(i=0; i<n-1; i++)
 { for(j=0; j<n-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)
   {
     sumo = sumo + a[i];
 }
```

```
}
Printf ("\n sum of odd index is %d", sum[illegible]);
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]);
    }
   }
}
```

```
5) #include<stdio.h>
   int recursive Binary search (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 Binary Search (array, start_index , middle -1,
                                              element ) ;
return resursive Binary search(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 Binary search (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;
 }
```