Aadarsh Reddy
AP19110010471
CSE-H

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 no of elements of array");
    Scanef("%d",&n);
    Printf("enter the element of array\n");
    for(i=0;i<n;i++)
        Scanf("%d",&a[i]);
    Sort(a,n);
    for(i=0;i<n+i;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\n", a[m1] + a[m2]);
Printf("the product is %d", a[m1] * a[m2]);
}
```

2) /* C program for merge Sort */

```
#include <stdlib.h>
#include <stdio.h>
// merge two subarray of arr[].
// First subarray is arr[l..m]
// Secondary subarray is arr[m+1..r]
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-1)/2;
   mergeSort(arr, l, m);
   mergeSort(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 = sizeof(arr)/sizeof(arr[0]);
   for(i=0; i<arr_size; i++){
       printf("enter the element");
       scanf("%d", &arr[i]);
   }
```

```
Printf("Given array is \n");
Print Array (arr, arr_size);
mergeSort (arr, 0, arr_size - 1);
Print("\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) 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.

Examples:-

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

// 25 12 22 64

// find the minimum elemet in arr[1..4]
// and place it at beginning of arr[1..4]

// 12 25 22 64

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

// Find the minimum element in arr[3..4]
// and place it at beginning of arr[3..4]
// 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 to 4

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 element in A[0..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 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, Semo = 0, proed = 1, m;
 Printf("Enter number of elements\n");
 Scanf ("%od", &n);
 Printf ("Enter %od 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)
   {
      Sumo= Sumo + a[i];
   }
}
Printf("In Sum of add Index is %d", Sumo);
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 recursive 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 = 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;
}
```