DSA ASSIGNMENT

Balaji.G
AP19110010517
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])
            {
                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 element's of a
   scanf("%d",&n);
   printf("enter the element's of array\n
   for(i=0;i<n;i++)
        scanf("%d",&a[i]);
   sort(a,n);
   for(i=0;i<n;j++)
        printf("%d",a[j]);
   printf("enter the element to find in
           array\n");
   scanf("%d",&e);
   f=binary(a,e,n);
   if(f!=0)
   {
        printf ("element is found at
                  %d position", f);
   }
  else
  {
      print ("element not found \n");
  }
 printf("enter the position of array
            to find sind 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<conio.h>
#define MAX-SIZE 5
Void merge-sort(int,int)
Void merge-array(int int int ,in
int arr-sort[Max-Size];
int main () {
  int i,k,Pro = 1;
  printf("Simple Merge Sort
          Example functions and
          Array\n");
  printf("\nEnter%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": Elements for sorting",
        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("\n Your Data :");
   for (i=0; i<MAX-SIZE; i++){
 }
  merge-sort(0, Max-Size-1);
  Printf("\t %d", arr-sort[i]);
}
Print("find the product of kth
      elements from first and last
      where k\n");
scanf("%d", &k);
```

```
Pru = arr-sort [k] * arr-sort [Max-size-k]
Printf ("Product = %d", Pru);
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-arrays (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++];
 else
  t[k++] = arr-sort[j++];
 }
//collect remaining element's
while (i<=b)
  t[k++] = area_sort[j++];
  for(i=a, j=0; i<=d; i++, j++)
  area-soent[i] = t[j];
}
```

3.)

[11:26 am].

The selection sort algorithm sort an array by repeatedly finding the minimum element (considering ascending order) from unsorted par and putting it at the begining. The algorithm maintains two subarray which is already sort.

arr [] = 64 25 12 22 11

// Find the minimum element in arr[0..4

// and place it at begining

11 25 12 22 64

//Find the minimum element 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]
// 12 22 25 64

//Find the minimum element in arr[3...4]
//and place it at beginning of arr[3...4]
// 12 22 25 64

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
insertion sort(arr, n)

Loop from i = 1 to n-1

a) Pick element arr[i] and insert it into sorted sequence arr[0...i-1]

[11 : 26 am

12, 11, 13, 5, 6

Let us loop for i = 1 (second element of the array) to 4 (last element of the array)

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 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>
voidmain()
{
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++)
```

```
{
  if(i%2!=0)
   {
    Sumo=Sumo+a(i);
   }
 }
Printf("\nsum of odd Index is %d
                              , sumo);
for(i=0;i<n;i++)
 {if (i%2==0)
   {
     Prod=Prod*a[i]
    }
  }
Printf("\nproduct of odd Index
                        is %d",prod
printf ("\nEnter 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 recursiveBinarySearch (int array [],
int start _Int end - index , int element){
if (end_index >= start_index){
int middle = start_index + (end_index
                              - start_index
```

```
if (array [middle] == element)
    return middle;
if (allay [middle] > element)

    returnrecurse Binary Seach(allay,
         start_index, middle-1, element);
Returnrecursive BinalySearch (allay,
middle+1, end_index, element);

}
return-1;
}
int main(void){
   int allay[] = {1,4,7, 9, 16, 56, 70
   int n = 7;
   int element = 9;
   int found_index = recursiveBinarySeac
           (allay, 0, n-1, element)
```

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