Assignment

A.V.S. Nishant
AP19110010362
CSE-F

1

```
#include <stdio.h>
#include <stdlib.h>
int des( );
int main ( );
{
        return des ( );
        return 0;
  }
  int des ( );
  {
      int arr[50],n,a,b,i,p,q,x,y,temp,first=0,
              last=n-1, mid, found=0;
     printf(" Enter the elements that you need to
                                        enter");
      scanf("%d",&n);
      for (a=0; a<n; a++)
     {
        printf("Enter the %dth element:",a);
         scanf("%d", & arr[a]);
      }
      for (a=0; a<n-1; a++).
      {
        for (b=0; b<n-a-1; b++)
        {
           if (arr[b]< arr[b+i])
           {
              temp= arr[b]
              arr[b] =arr[b+1];
              arr[b+1]= temp;
           }
        }
      }
```

```
printf("Sorted list indescending order is:\n");
for (a=0; a<n; a++)
{
   printf("%d\n", arr[a]);
}
 printf("Enter the element that you need to
                                   search");
 scanf("%d", &key);
 while (first <= last && !found)
 {
    mid = (first+last)/2;
    if (arr (mid ]==key)
    {
       found=1;
    }
    else if (arr[mid] > key)
    {
       last= mid-1;
    }
    else
    {
       find= mid+1;
    }
    if (found)
    {
       return mid;
    }
    else
    {
       return -1
    }
    if (found == 1)
    {
       printf("%d is found in the locatio
                           %d, key, mid);
}
else
 {
  printf(%d is not found in the array",kg);
 }
}
printf("Enter the 1st location:");
scanf("%d",&x);
printf("Enter the 2nd location:");
scanf("%d",&y);
if (x>n || y>n)
 {
  printf("please enter the valid location");
 }
 else
 {
  p=arr[x]+arr[y]
  printf("Sum of value in the locatois%d,
                                          p);
  q=arr[x]*arr[y];
  printf("product of values in the locations
                          %d\n,q);
 }
}
```

```
② #include <stdio.h>
void mergesont (int a[ ], int i, int j );
void merge(int a[], int i1, int j1, int i2, int j2);
{
  int temp[50]; i, j, k,
   i= i1
   j = i2
   k =0;
  while ( i<= j1&& j<= j2)
  {  if (a[i]< a[j])
        temp [k++]= a [j++];
   }
   while (i<= j1)
     temp[k++]=a[i++];
    while ( j <= j2)
       temp (k++]=a[j++];

   for (i =i1, j=0; i<=j2; i++ ,j++)
      a[i]= temp[j],

  int main ()
  {
    int a[50],n,i;
    printf ("Enter no. of elements");
    scanf ("%d",&n);
    printf ("Enter array elements");

    for (i=0, i<n; i++)
     scanf ("%d", & a[i]);
merge sort (a,0,n-1);
printf("%d",a[i]);
return 0;
}
void merge sort (int a[],int i,int j)
{
  int mid;
  if (i<j)
  {
    mid = (i+j)/2
    mergesort (a, i, mid);
     merge sort( a, mid+1, j );
     merge (a, i, mid, mid+1, j);
  }
}
```

(3) **Selection Sort** :- The selection sont algorithum sorts an away by repeatedly finding the minimum element from unsonted part and putting it at the beginning. The algorithm maintainy two subarrays in a given away.

1) The subarray which is alredy sonted
2) Remaing subarray which is unsonted.

In every interation of selection sont, the minimim element from the unsonted subarray is picked and moved to tee sonted 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

// Find the minimum element in arr[0...4]
// and place it at beggiing of arr[0...4]

11 12 25 22 64

// Find the minimen elent in arr[2...4]
// place it at begining of arr[2...4]

11 12 22 25 64

// find the minimu elemet in arr[3..4]
// place it at beginig of arr[3...4]

11 12 22 25 64

Insention Sont :- It is a simple sonting algorithum that wonks the way we sont playu cards in our hands.

Algorithm

```
//sont an arr[ ] of size n
  insention sont (arr, n)
    loop from i = 1 to n-1.
  a) Pick elemt arr[i] and insent it into sonted
     sequene arr[0...i-1]
```

Example:

12 , 11 , 13 , 5 , 6

let us loop from i=1 (2^n elent of array) to i=4 (last element of array.

i= 1. since 11 is smaller than 12, move 12 and insent 11 before 12

11 , 12, 13, 5, 6

i=2. 13 will remain at its position as all elements are smaller than 13

11, 12, 13, 5, 6.

i = 3. 5 will move to the starting and all other elemnt from 11 to 13 will move one position ahead of the current position.

5, 11, 12, 13, 6

i=4. 6 will move to after 5. and all the elements from 11 to 13 will move a position ahad of the current position.

Scanned with CamScanner

```
(4) # include< stdlib.h>
int main( )
{
 int array[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 ,&array[i]);
  for ( i=0, i<n-1, i++ )
  { for (j=0, i<n-i-1; j++)
   {
   if (array[j]> array[j+1]) /~~for~~
    {
      temp = array[j]
       array[j]= array[j+1];
        array [j +1] = temp;
    }
   }
  }
  printf ("Sorted list in ascending order \n");
  for (i=0; i<n, i++)
  printf ("%d \n", array[i]);
  printf ( "Sorted list in alternated order:\n)
  for (i= 0, i<n , i= i+2)
```

```
printf ("%d\n", array[i ]);
printf ("Sum of all the elements in odd posi

for (i=0, i<n, i=i+2)
 Sum= Sum + array[ i ]
  printf ("% d\n", Sum; ).
  printf (" Product of all the elemets in ev
                              position\n

 for (i=1, i<n, i=i+2 )
  prod = prod * array[i];
  printf ("%d\n", prod );
  printf (" Enter a number \n");
  Scanf ("%d', &m);
  printf (" Element divisible by %d ar
                                \n",

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

(5)

```
# include <stdio.h>
int recurive Binarysearch( int array[ ], int start_ind,
                int end_index, int element)
{
  if (end_index >= start_index)
  {
    int middle = start_index + (end_index - start_ind)
                                              /2;
    if (array[ middle ] > element)
    return recurive Binarysearch(array, start_ind,
                               middle-1 ,element);
    return recurive 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 = recurive Binary Search
                             (array,0,n-1,element,
    if (found_index = = -1)
    {
     printf ("Elemet not found in array");
    }
    else
    {
     printf ("Elenet found in index%d");
    }
} return 0;
```