Souvik Mandal
AP19110010485
CSE - F

Assignment - 6

1) a)
```
# include < stdio.h>
void main()
{
 int number [30];
 int i, j, a, n;
 printf ("enter the value of N\n");
 scanf( "%d", &n);
 printf ( "Enter the numbers \n");
 for ( i=0; i<n; ++i)
 scanf( "%d", & number [ i]);
 ~~for ( j = i+1; j<n; ++j)~~
 for ( i = 0; i<n; ++i)
 { for ( j = i+1; j<n; ++j) {
  if ( number [ i]< number [j]
  {
   a = number [i];
   number [i] = number[ j];
   number [j] = a;
  }
 }
}

printf(" The numbers in descending order \n");
for( i=0; i<n; ++i)
{
 printf("%d", number [i]);
}
int first, last, middle, search, array[100];
printf("Enter value to find \n");
 scanf("%d", &search);
  first = 0;
  last = n - 1;
  middle = (first + last) / 2;
  while (first <= last)
  {
   if (array[middle] < search)
     first = middle + 1;
   else if (array[middle] == search)
   {
   printf("%d found at location %d \n", search, middle);
   break;
   }
   else
    last = middle - 1;
   middle = (first + last)/2
   }
   if (first > last)
   printf(" Not found !", search);
   return 0;
   }
```

b)
```
  # include < stdio.h>
    void main()
    {
     int number[30]
     int i, j, a, n;
     printf ("enter the value");
     scanf ("%d", &n);
     printf ("Enter the numbers");
     for (i=0; i<n; ++i)
         scanf ("%d", &number[i]);
    for (i=0; i<n; ++i)
     {
      for (j=i+1; j<n; ++j)
      {
        if ( number [i] < number [j])
      {
       a = number [i];
      number [i] = number [j];
       number [j] = a;
      }
     }
    }
     printf ("numbers arranged in decending order");
    for (i=0; i<n; ++i)
     {
```

```
printf("%d", number [i]);
}

{ int m1 , m2 ;
printf(" Enter the position of array to find sum and product \n");
scan (" %d %d", & m1 , & m2);
  m1 --;
  m2 --;
printf(" The sum is %d", number [m1] + number [m2]);
printf(" The product is %d", number [m1] * number [m2]);
}
return 0;
}
```

2)
```
# include <stdlib.h>
# include <stdio.h>
 void merge [int arr [], int l, int m , int r)
 {
  int i, j, k;
  int n1 = m - l + 1 ;
  int n2 = r - m ;
  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 = I;
while (i<n, && j< n2)
{
 if (L[i] <= R[j])
  {
   arr[k] = L[i]
    i++;
   }
  else
  {
   arr[k] = R[j];
   j++;
  }
else
{ arr[k] = R[j];
  j ++;
  }
  k ++
}
 while (j < n2)
 {
  arr(k) = R[j]
   j++;
   k++;
  }
```

```
}
void merg ~~sot~~ sort (int arr[.], int I, int r)
{
if (I<r)
 {
  int m = I +(r -1) /2
  merg sort (arr, I, m);
  merge sort (arr, m+1, r);
  merge ( arr, I, m, r);
  }
 }
 void print array ( int A [], int size)
  {
  int i ;
  for (i=0; i<size ; i++)
     printf( "%d", A[i]);
     printf("\n");
  }
  int main()
  {
   int size, v ;
   printf ("enter array size :");
   scanf ("%d", & size);
    int vab [size];
    for ( v=0; v< size ; v++)
```

```
{
 printf ("Enter value : ");
 scanf ("%d" ; & val [v]);
}
 printf ("Given array is ");
 print ~~A~~ Array (val, size);
 merge sort (val, size - 1);
 printf (" sorted array");
 printArray ( val , size);
 int k, j , P1 , P2 , temp ;
 printf (" Enter the value of k ");
 scanf ("%d", & k)
     P1 = P2 = 1 ;

 for ( j = 0; j <= k ; j++)
  {
   temp = val [j] ;
   P 1* = temp ;
  }
 for ( I = size - 1 ; I >= k ; I--)
  {
   temp = val [ I ] ;
   P 2* = temp ;
  }
```

Scanned with CamScanner

```
}
printf(" product of elements : %d %d", P1 , P2);
}
```

3) Insertion sort :-

Insertion is a process of one type of sorting which works by inserting the set values in the existing sorted file. This works by inserting a single ~~s~~ element at a time. This process continious Until whole array is sorted in same order. This method save an effective amount of memory.

~~Advantages~~ Advantages of Insertion sort :-

* The memory space takes less.
* It is faster than any other algorithm.
* Easy to impliment.

Selection sort :-

This sort perform by searching for the minimum value number and placing it into the first position according to the order.

Advantages of selection sort :-

* suppose an array with N elements in the memory
* simple to understand the sorting of elements doesn't depend on initial arrangment of the elements.

4) i)

```
# include <stdio.h>
int main ()
{
int arr [100], n, c, d; swap;
printf ("Enter number of elements :");
scanf ("%d", & n);
printf (" Enter %d integers :", n);
for ( c=0; c<n ; c++)
scanf ("%d", & array [c]);
for ( c=0; c<n-1; c++)
{
 for (d=0; d<n-c-1; d++)
{
 if (array [d]> ~~array~~ array [d+1])
 {
  swap = array [d];
  array[d] = array [d +1];
  array [d+1] = swap;
 }
```

```
  }
 }
printf(" sorting in accending order:");
for (c=0; c<n; c++)
  printf("%d \n", array[c]);
printf("Alternate elements of array:");
for (n=0; n<10; n+=2)
  printf("%d, \n", array[n]);
  return 0;
}
```

ii) adding to the program $\boxed{4}$ (i)

```
{
 int sum, product;
 for (c=0; c<n; c+=2)
 {
   sum = sum + array[c];
  }
  for (c=1; c<n; c+=2)
  {product = product * array[c];
  }
   printf("sum : %d \n", sum);
  printf(" product : %d \n", product);
  return 0;
```

```
(iii) {
   int m ;
   Printf("Enter value m");
   scanf("%d", &m);
   printf("numbers divisible by %d are :", m);
   for(c=0; c<n; c++)
   {
   if(array[c]%m=0)
   {
    printf("%d\t", array[c]);
   }
   return 0 ;
   }

5)  # include <stdlib.h>
    # include <stdio.h>
      int bin search( int[a], int l, int h, int x)
      {
       int mid =(l+h)/2;
        if( l>h) return -1;
       if (a[mid] ==x)
       return mid ;
        if [a [mid] <x)
       return bin search (a, mid +l , h, x);
       else
       return bin search ( a, l  mid +1,  h, x);
      }
```

```
int main (void)
{
 int a[100];
 int size, pos, val;
 printf ("enter length of array:");
 scanf ("%d", & size);
printf ("\n Enter array elements");
for ( int i=0; i<size; i++)
 scanf ("%d", & a[i]);
printf ("Enter element to search:");
scanf ("%d", & val];
 pos = binsearch ( a, 0, size-1, val);
if (pos<0)
 {
 printf ("can't find the element %d in array", val);
 }
 else
 {
 printf ("position of %d in array is %d \n", val, pos+1);
 }
 return 0;

 }
```