**DS FILE**

**Lab : 1 - 11**

**NAME – Adarsh Chaudhary**

**ROLL NO. – 2100320120007**

**BRANCH – CS**

**SECTION - A**

**PROGRAM 1A - Program for Insertion in any array**

**ALGORITHM Insertion(A[], N, i, key)**

**BEGIN:**

FOR j=N TO i STEP-1 DO

A[j+1]=A[j]

A[i]=key

N=N+1

**END;**

**Time Complexity:ϴ(N)**

**Space Complexity:ϴ(1)**

#include<stdio.h>

int main()

{

printf("This code is written by ABHAY PANDEY //CS-A//2100320120004 \n");

    int arr[100];

    int n;

    printf("Enter the size of array\n");

    scanf("%d",&n);

    printf("Enter %d elements\n",n);

    for(int i=0;i<n;i++)

    scanf("%d",&arr[i]);

    int posi;

    printf("position:");

    scanf("%d",&posi);

    int ele;

    printf("Enter element\n");

    scanf("%d",&ele);

    for(int i=n;i>=posi;i--)

        arr[i]=arr[i-1];

        arr[posi-1]=ele;

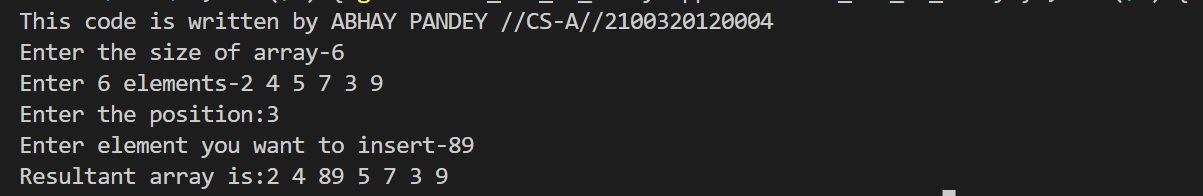
        n++;

    for(int i=0;i<n;i++)

        printf("%d",arr[i]);

    return 0 ;}

**OUTPUT:**



**PROGRAM 22 - Transpose without using second matrix**

**ALGORITHM: Matrixtranspose(A[][], M,N)**

**BEGIN:**

FOR i=1 TO M DO

FOR j=1 TO i DO

temp=A[i][j]

A[i][j]=A[j][i]

A[j][i]=temp

RETURN A

**END;**

**Time Complexity: ϴ(N2)**

**Space Complexity:ϴ(1)**

#include<stdio.h>

int main()

{

printf("This code is written by ABHAY PANDEY //CS-A//2100320120004 \n");

int n,m;

    printf("Enter the rows and columns of matrix:\n");

    scanf("%d%d",&n,&m);

    int arr[n][m];

    printf("Enter the elements of matrix:\n");

     for(int i=0;i<n;i++)

     {

        for(int j=0;j<m;j++)

            scanf("%d",&arr[i][j]);

     }

    for(int i=0;i<n;i++)

    {

        for(int j=i;j<m;j++)

        {

            int temp=arr[i][j];

            arr[i][j]=arr[j][i];

            arr[j][i]=temp;

        }

    }

    printf("Transpose of the matrix is:\n");

     for(int i=0;i<m;i++)

     {

        for(int j=0;j<n;j++)

        printf("%d",arr[i][j]);

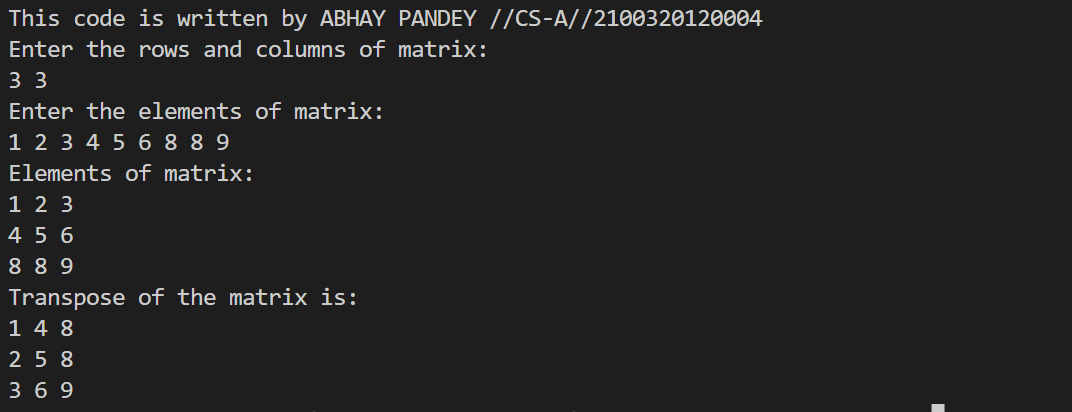
        printf("\n");

     }

    return 0 ;

}

**Output:**



**PROGRAM 1C - Program for Traversing of array**

**ALGORITHM Traverse(A[], N)**

**BEGIN:**

FOR i=1 TO N DO

WRITE(A[i])

**END;**

**Time Complexity:ϴ(N)**

**Space Complexity:ϴ(1)**

#include<stdio.h>

int main()

{

printf("This code is written by ABHAY PANDEY //CS-A//2100320120004 \n");

    int n;

    printf("Enter the size of array:");

    scanf("%d",&n);

    int arr[n];

    printf("Enter the elements of array:");

    for(int i=0;i<n;i++)

    scanf("%d",&arr[i]);

    printf("Elements of array are-->\n");

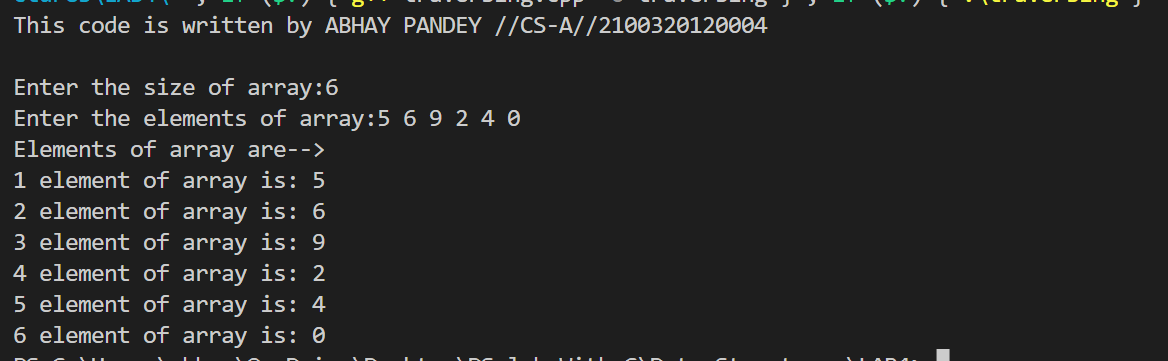
    for(int i=0;i<n;i++)

    printf("%d element of array is: %d\n",i+1,arr[i]);

    return 0 ;

}

**OUTPUT:**



**PROGRAM 1B - Program for Deletion of elements in array**

**ALGORITHM Deletion(A[], N, i)**

**BEGIN:**

X=A[i]

FOR j=i+1 TO N DO

A[j-1]=A[i]

N=N-1

RETURN x

**END;**

**Time Complexity:ϴ(N)**

**Space Complexity:ϴ(1)**

#include <stdio.h>

int main()

{ printf("This code is written by ABHAY PANDEY //CS-A//2100320120004 \n");

int array[100], position, c, n;

printf("Enter number of elements in array\n");

scanf("%d", &n);

printf("Enter %d elements\n", n);

for ( c = 0 ; c < n ; c++ )

scanf("%d", &array[c]);

printf("Enter the location where you wish to delete element\n");

scanf("%d", &position);

if ( position >= n+1 )

printf("Deletion not possible.\n");

else

{

for ( c = position - 1 ; c < n - 1 ; c++ )

array[c] = array[c+1];

printf("Resultant array is\n");

for( c = 0 ; c < n - 1 ; c++ )

printf("%d\n", array[c]);

}

return 0; }

**Output:**

Text

Description automatically generated

**PROGRAM 3 - Program to Find the number, which is not repeated in Array of integers, others are present for two times**

**ALGORITHM: Arr\_func(A[], N)**

**BEGIN:**

K=0,c,B[20]

FOR i=0 TO N DO

c=0

FOR j=0 TO N DO

IF A[j]==A[i] THEN

c=c+1

IF c==1 THEN

B[k++]=A[i]

FOR i=0 TO k DO

WRITE(B[i])

**END;**

**Time Complexity:ϴ(N2)**

**Space Complexity:ϴ(1)**

#include<stdio.h>

void unique(int arr[],int n)

{

    int count=1,i,j;

    for( i=0;i<n;i++)

    {

        for( j=0;j<n;j++)

        {

            if(arr[i]==arr[j]&& i!=j)

            break;

        }

            if(j==n)

            {

                printf("Unique element %d is:%d\n",count,arr[i]);

                count++;

             }

    }

}

int main()

{ printf("This code is written by ABHAY PANDEY //CS-A//2100320120004 \n");

    int n;

    printf("Enter size of array:\n");

    scanf("%d",&n);

    int arr[n];

    printf("Enter array elements:\n");

    for(int i=0;i<n;i++)

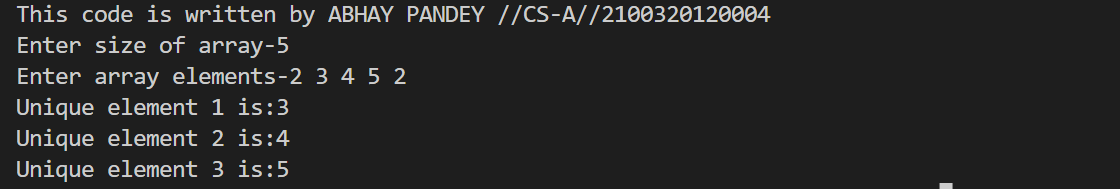
    scanf("%d",&arr[i]);

    unique(arr,n);

    return 0 ;

}

**OUTPUT:**



**PROGRAM 63 - Program for finding nth Fibonacci number using Recursion and improving its run time to save stack operations**

**ALGORITHM** Fibo(a)

BEGIN:

IF a==1 THEN

RETURN 0

ELSE

IF a==2 THEN

RETURN 1

ELSE

RETURN Fibo(a-1)+Fibo(a-2)

END;

**Time Complexity: ϴ (2N)**

**Space Complexity: ϴ(N)**

#include<stdio.h>

int fibo(int n){

if(n<=1)

return n;

return fibo(n-1)+fibo(n-2);

}

int main()

{ printf("This code is written by ABHAY PANDEY //CS-A//2100320120004 \n");

int n;

printf("Enter the number:");

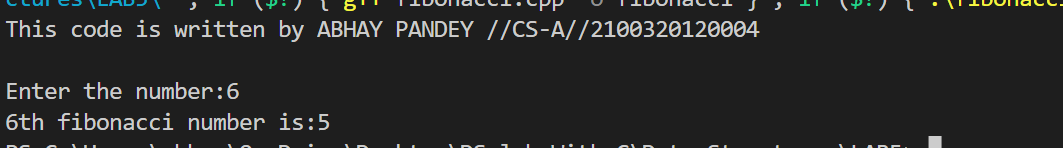
scanf("%d",&n);

printf("%dth fibonacci number is:%d",n,fibo(n-1));

return 0 ;

}

**Output:**



**PROGRAM 59 - Program for** **factorial of a given number using recursion**

**ALGORITHM** FACTORIAL(a)

BEGIN :

IF a==0

RETURN(1)

ELSE

IF(a>0)

RETURN(a\*FACTORIAL(a-1))

END;

**Time Complexity: ϴ(n)**

**Space Complexity: ϴ(n)**

#include <stdio.h>

#include<math.h>

int fact(int n){

if (n==0)

{

return 1;

}

else

{

return n \* fact(n-1);

}

}

int main(){

printf("This code is written by ABHAY PANDEY //CS-A//2100320120004 \n");

int n;

printf("Enter the number : \n");

scanf("%d",&n);

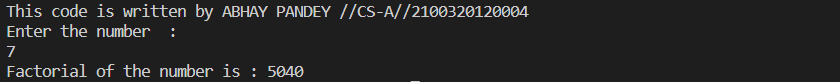
printf("Factorial of the number is : ");

printf("%d",fact(n));

return 0;

}

**Output:**

**PROGRAM 64 - Program for finding the GCD of two numbers using Recursion**

ALGORITHM HCF(a,b)

BEGIN:

IF a==b THEN

RETURN a

ELSE IF a>b THEN

RETURN HCF(a-b,b)

ELSE

RETURN HCF (a,b-a)

END;

**Time Complexity: O(log n)**

**Space Complexity: ϴ(1)**

#include <stdio.h>

#include <math.h>

int gcd(int a, int b)

{

if (a == b)

{

return a;

}

else

{

if (a > b)

{

return gcd(a - b, b);

}

else

{

return gcd(a, b - a);

}

}

}

int main()

{ printf("This code is written by ABHAY PANDEY //CS-A//2100320120004 \n");

int a, b;

printf("Enter the numbers : \n");

scanf("%d %d", &a, &b);

printf("GCD of the numbers is : ");

printf("%d", gcd(a, b));

return 0;

}

**Output:**

Shape

Description automatically generated with medium confidence**EXPERIMENT 61 - Program for Computing A raised to power n using Recursion**

ALGORITHM POWER(a,b)

BEGIN:

IF b == 0 THEN

RETURN 1

ELSE

IF b%2 == 0 THEN

RETURN POWER(a,b/2) \* POWER(a,b/2)

ELSE

RETURN a+ POWER(a,b/2) \* POWER(a,b/2)

END;

**Time Complexity: O(log b)**

**Space Complexity: ϴ(log b)**

#include <stdio.h>

#include <math.h>

int power(int a, int b)

{

if (b == 0)

{

return 1;

}

else

{

return a \* power(a, b - 1);

}

}

int main()

{ printf("This code is written by ABHAY PANDEY //CS-A//2100320120004 \n");

int a, b;

printf("Enter the numbers : \n");

scanf("%d %d", &a, &b);

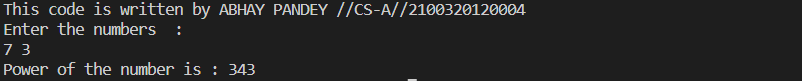
printf("Power of the number is : ");

printf("%d", power(a, b));

return 0;

}

**Output:**

**PROGRAM 65 - Program to reverse the given number using Recursion**

**ALGORITHM** REV (a,len)

BEGIN:

IF len ==1

RETURN a

ELSE

RETURN((a%10)\*pow(10,len-1))+REV(a/10,len-1)

END;

**Time Complexity: ϴ (log n)**

**Space Complexity: ϴ (log n)**

#include <stdio.h>

#include<math.h>

int reverse(int n,int temp,int sum)

{

if (n > 0)

{

temp = n % 10;

sum = sum \* 10 + temp;

reverse(n / 10 , temp,sum);

}

else

{

return sum;

}

}

int main()

{ printf("This code is written by ABHAY PANDEY //CS-A//2100320120004 \n");

int n;

int temp = 0, sum = 0;

printf("Enter the number : ");

scanf("%d",&n);

printf("Reverse of the number is : ");

printf("%d", reverse(n,temp,sum));

return 0;

}

**Output:**

**PROGRAM 60 - Program for Towers of Hanoi for n disk (user defined)**

ALGORITHM TOH(N,S,M,D)

BEGIN:

IF N==1 THEN

Transfer disk from S to D

ELSE

TOH(N-1,S,M,D)

Transfer Disk From S to D

TOH(N-1M,S,D)

End;

**Time Complexity: ϴ (2n)**

**Space Complexity: ϴ (n)**

#include <stdio.h>

#include<math.h>

void tower\_of\_hanoi(int n,int s,int m,int d){

if (n>0)

{

tower\_of\_hanoi(n-1,s,d,m);

printf("Move from %d -> %d \n",s,d);

tower\_of\_hanoi(n-1,m,s,d);

}

}

int main(){

printf("This code is written by ABHAY PANDEY //CS-A//2100320120004 \n");

int n;

printf("Enter the number of discs : ");

scanf("%d",&n);

printf("Process to transfer discs are :");

tower\_of\_hanoi(n,1,2,3);

return 0;

}

**Output:**

**Text

Description automatically generated**

**PROGRAM 2 - Program for Insertion in sorted array**

**ALGORITHM Sorted(A[], N, key)**

**BEGIN:**

i=0

WHILE A[i]<key DO

i=i+1

RETURN i

**END;**

**Time Complexity:ϴ(N)**

**Space Complexity:ϴ(1)**

**ALGORITHM: INS\_sorted(A[], N ,i, key)**

**BEGIN:**

FOR j=N-1 TO i STEP-1 DO

A[j+1]=A[j]

A[i]=key

N=N+1

**END;**

**Time Complexity:ϴ(N)**

**Space Complexity:ϴ(1)**

#include<stdio.h>

int main()

{ printf("This code is written by ABHAY PANDEY //CS-A//2100320120004 \n");

    int n;

    printf("Enter the size of array:\n");

    scanf("%d",&n);

    int arr[n];

    printf("Enter the array elements:");

        for(int i=0;i<n;i++)

        {

        scanf("%d",&arr[i]);

    }

   int ele;

   printf("Enter the element that you wants to enter:");

   scanf("%d",&ele);

   int pos=0;

   for(int i=0;i<n;i++)

   {

    if(arr[i]<ele)

        pos++;

    else

        break;

   }

   for(int i=n;i>=pos;i--)

        arr[i]=arr[i-1];

        arr[pos]=ele;

        n++;

    printf("Array after the insertion is:\n");

    for(int i=0;i<n;i++){

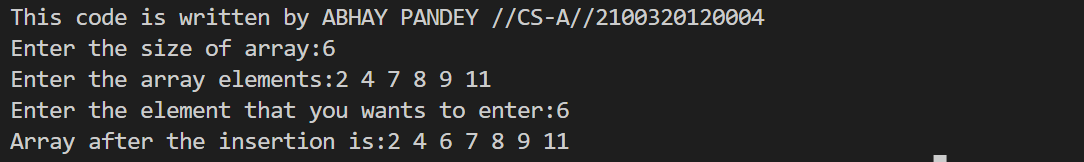
        printf("%d",arr[i]);

    }

    return 0;

}

**OUTPUT:**



**PROGRAM 15 - Program for Intersection of two Sets**

**ALGORITHM: SetIntersection(A[],m,B[],n)**

**BEGIN:**

C[m+n]

i=1, j=1, k=1

WHILE i<=m AND j<=n DO

IF A[i]<B[j] THEN

i=i+1

ELSE

IF A[i]==B[j] THEN

C[k]=B[j]

i=i+1

j=j+1

k=k+1

ELSE

j=j+1

RETURN C

**END;**

**Time Complexity:ϴ(N)**

**Space Complexity:ϴ(N)**

#include<stdio.h>

void intersection(int arr[],int brr[],int n,int m)

{

    int i=0,j=0;

    printf("Instersection of first and second set is:");

    while(i<n and j<m)

    {

        if(arr[i]<brr[j])

            i++;

        else if(arr[i]>brr[j])

            j++;

        else

        {

            printf("%d ",arr[i]);

            i++;

            j++;

        }

    }

}

int main()

{ printf("This code is written by ABHAY PANDEY //CS-A//2100320120004 \n");

    int n,m;

    printf("Enter the size of first and second set :");

    scanf("%d%d",&n,&m);

    int arr[n],brr[m];

    printf("Enter the first set elements:");

    for(int i=0;i<n;i++)

    scanf("%d",&arr[i]);

    printf("Enter the second set elements:");

    for(int j=0;j<m;j++)

    scanf("%d",&brr[j]);

    // sort(arr,arr+n);

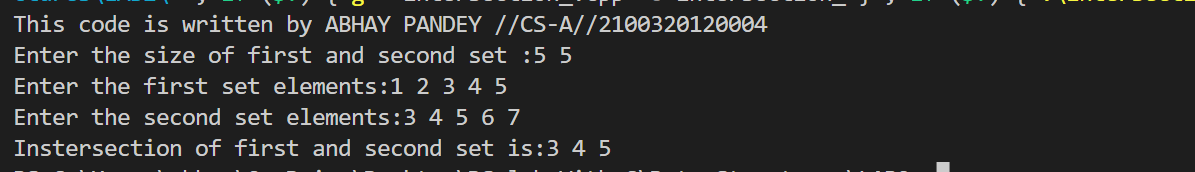
    // sort(brr,brr+m);

    intersection(arr,brr,n,m);

    return 0 ;

}

**output:**



**PROGRAM 11 - Program for Merging of two Sorted arrays**

**ALGORITHM: MergeArr(A[],m,B[],n)**

**BEGIN:**

C[m+n]

i=1, j=1, k=1

WHILE i<=m AND j<=n DO

IF A[i]<B[j] THEN

C[k]=A[i]

i=i+1

k=k+1

ELSE

C[k]=B[j]

J=j+1

k=k+1

WHILE i<=m DO

C[k]=A[i]

i=i+1

k=k+1

WHILE j<=n DO

C[k]=B[j]

J=j+1

k=k+1

RETURN C

**END;**

**Time Complexity:** 𝚹**(N)**

**Space Complexity:** 𝚹**(N)**

#include<stdio.h>

void merge(int arr[],int brr[],int n,int m,int ans[])

{

   int i=0,j=0,k=0;

   printf("Sets after the merging is:");

   while(i<n&&j<m)

   {

    if(arr[i]<brr[j])

    ans[k++]=arr[i++];

    else

    ans[k++]=brr[j++];

   }

    while(i<n)

    ans[k++]=arr[i++];

    while(j<m)

    ans[k++]=brr[j++];

    for(int i=0;i<n+m;i++)

printf("%d ",ans[i]);

   }

int main()

{ printf("This code is written by ABHAY PANDEY //CS-A//2100320120004 \n");

int n,m;

    printf("Enter the size of first and second set:");

    scanf("%d%d",&n,&m);

    int arr[n],brr[m];

    printf("Enter the first set elements:");

    for(int i=0;i<n;i++)

    scanf("%d",&arr[i]);

    printf("Enter the second set elements:");

    for(int j=0;j<m;j++)

    scanf("%d",&brr[j]);

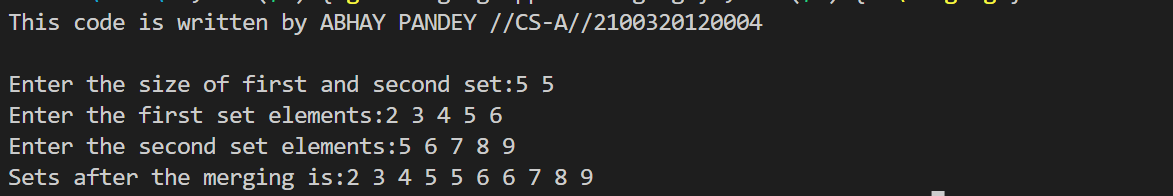
    int ans[n+m];

    merge(arr,brr,n,m,ans);

    return 0 ;

}

**OUTPUT:**



**PROGRAM 16 - Program for Set Difference**

**ALGORITHM: SetDIFference(A[],m,B[],n)**

**BEGIN:**

C[m+n]

i=1, j=1, k=1

WHILE i<=m AND j<=n DO

IF A[i]<B[j] THEN

i=i+1

ELSE

IF A[i]==B[j] THEN

i=i+1

j=j+1

ELSE

C[k]=B[j]

j=j+1

k=k+1

WHILE j<=n DO

C[k]=B[j]

J=j+1

k=k+1

RETURN C

**END;**

**Time Complexity:ϴ(N)**

**Space Complexity:ϴ(N)**

#include<stdio.h>

void AminusB(int arr[],int brr[],int n,int m){

    int k=0;

    int ans[100];

    int i,j;

    printf("Difference of both sets(i.e, A-B) is:");

    for(i=0;i<n;i++)

    {

        for(j=0;j<m;j++)

        {

            if(arr[i]==brr[j])

            break;

        }

            if(j==m)

            ans[k++]=arr[i];

    }

    for(int i=0;i<k;i++)

    printf("%d ",ans[i]);

}

void BminusA(int arr[],int brr[],int n,int m){

int k=0;

    int ans[100];

    int i,j;

    printf("Difference of both sets(i.e, B-A) is:");

    for(i=0;i<m;i++){

        for(j=0;j<n;j++)

        {

            if(brr[i]==arr[j])

            break;

        }

        if(j==n)

         ans[k++]=brr[i];

    }

    for(int i=0;i<k;i++)

    printf("%d ",ans[i]);

}

int main()

{  printf("This code is written by ABHAY PANDEY //CS-A//2100320120004 \n");

int n,m;

    printf("Enter the size of A and B set:");

    scanf("%d%d",&n,&m);

    int arr[n],brr[m];

    printf("Enter the set A elements:");

    for(int i=0;i<n;i++)

    scanf("%d",&arr[i]);

    printf("Enter the set B elements:");

    for(int j=0;j<m;j++)

    scanf("%d",&brr[j]);

    int i=0;

    int j=0;

    int c;

    printf("Enter the choice-\n1 for A-B\n2 for B-A\n");

    scanf("%d",&c);

    if(c==1)

    AminusB(arr,brr,n,m);

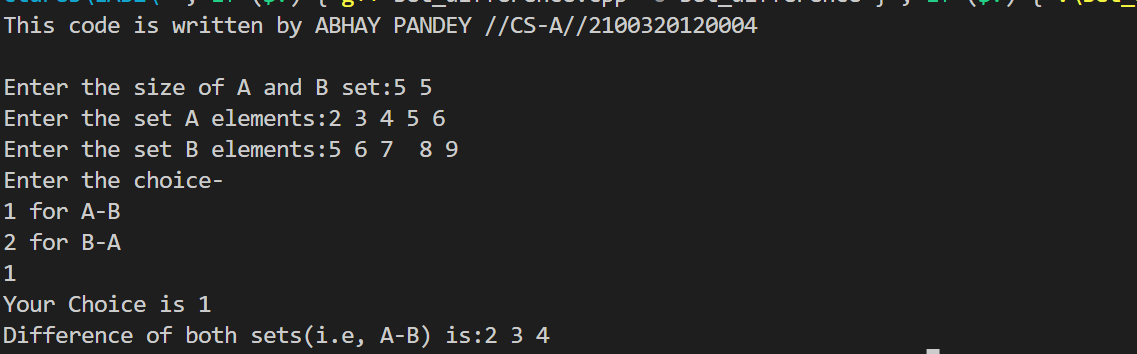
    if(c==2)

    BminusA(arr,brr,n,m);

    return 0 ;

}

**Output:**



**PROGRAM 14 - Program for Union of two sets**

**ALGORITHM: SetUnion(A[],m,B[],n)**

**BEGIN:**

C[m+n]

i=1, j=1, k=1

WHILE i<=m AND j<=n DO

IF A[i]<B[j] THEN

C[k]=A[i]

i=i+1

k=k+1

ELSE

IF A[i]==B[j] THEN

C[k]=B[j]

i=i+1

j=j+1

k=k+1

ELSE

C[k]=B[j]

j=j+1

k=k+1

WHILE i<=m DO

C[k]=A[i]

i=i+1

k=k+1

WHILE j<=n DO

C[k]=B[j]

J=j+1

k=k+1

RETURN C

**END;**

**Time Complexity:ϴ(N)**

**Space Complexity:ϴ(N)**

#include<stdio.h>

void unionArr(int arr[],int brr[],int n,int m,int ans[])

{

   int i=0,j=0,k=0;

   while(i<n&&j<m)

   {

    if(arr[i]<brr[j])

    ans[k++]=arr[i++];

    else if(arr[i]=brr[j])

    {

        ans[k++]=arr[i++];

        j++;

}

    else

    ans[k++]=brr[j++];

   }

    while(i<n)

    ans[k++]=arr[i++];

    while(j<m)

    ans[k++]=brr[j++];

    printf("Union of the first and second set is:");

    for(int i=0;i<k;i++)

    printf("%d ",ans[i]);

}

int main()

{ printf("This code is written by ABHAY PANDEY //CS-A//2100320120004 \n");

    int n,m;

    printf("Enter the size of first and second set :");

    scanf("%d%d",&n,&m);

    int arr[n],brr[m];

    printf("Enter the first set elements:");

    for(int i=0;i<n;i++)

    scanf("%d",&arr[i]);

    printf("Enter the second set elements:");

    for(int j=0;j<m;j++)

    scanf("%d",&brr[j]);

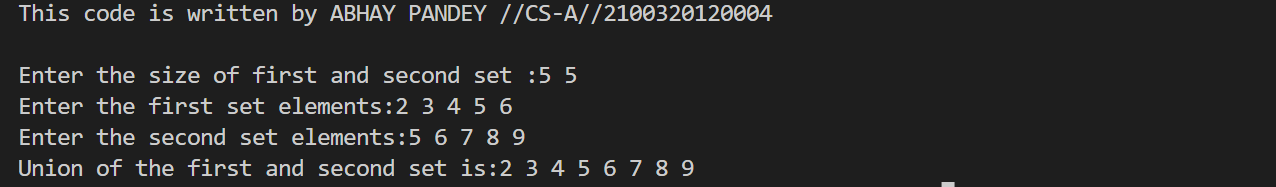
    int ans[n+m];

    unionArr(arr,brr,n,m,ans);

    return 0 ;

}

Output:



**PROGRAM 5 - Program for Binary Search in an array**

**ALGORITHM Binary\_search(A[], N, key)**

**BEGIN:**

HIGH=N-1

LOW=0

WHILE LOW<=HIGH DO

MID=(LOW+HIGH)/2

IF A[MID]==key THEN

RETURN MID

ELSE

IF key<A[MID] THEN

HIGH=MID-1

ELSE

LOW=MID+1

RETURN -1

**END;**

**Worst Case Time Complexity: O(logN)**

**Best Case Time Complexity: 𝞨(1)**

**Space Complexity: ϴ(1)**

#include<stdio.h>

int binarySearch(int arr[],int n,int key){

    int s=0;

    int l=n;

     while(s<=l)

     {

        int mid=(s+l)/2;

        if(arr[mid]>key)

        l=mid-1;

        else if(arr[mid]<key)

        s=mid+1;

        else

        return mid;

    }

    return -1;

}

int main()

{ printf("This code is written by ABHAY PANDEY //CS-A//2100320120004 \n");

    int n;

    printf("Enter the size of array:");

    scanf("%d",&n);

    int arr[n];

    printf("Enter the elements of array:");

    for(int i=0;i<n;i++)

    scanf("%d",&arr[i]);

    int key;

    printf("Enter the element to search:");

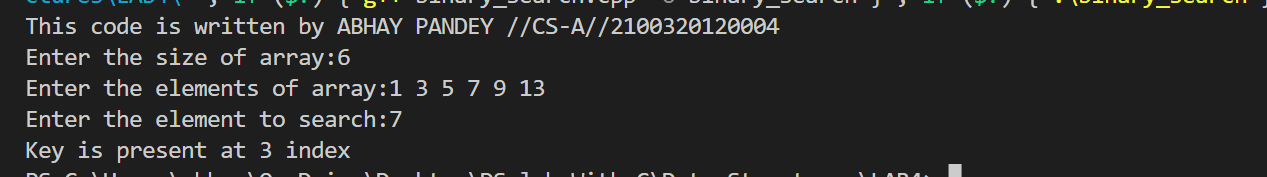
    scanf("%d",&key);

    printf("Key is present at %d index",binarySearch(arr,n,key));

    return 0 ;

}

**Output:**

**PROGRAM 4 - Program for Linear Search**

**ALGORITHM Linear\_search(A[], N, key)**

**BEGIN:**

FOR i=1 TO N DO

IF A[i]==key THEN

RETURN i

RETURN -1

**END;**

**Worst Case Time Complexity: O(N)**

**Best Case Time Complexity: 𝞨(1)**

**Space Complexity: ϴ(1)**

#include<stdio.h>int main()

 { printf("This code is written by ABHAY PANDEY //CS-A//2100320120004 \n");

    int n;

    printf("Enter the size of array: ");

    scanf("%d",&n);

     int arr[n];

     printf("Enter the elements of array :");

     for(int i=0;i<n;i++)

     scanf("%d",&arr[i]);

  int key;

    printf("Enter the element to be search:");

    scanf("%d",&key);

    int flag=0;

    for(int i=0;i<n;i++)

    {

        if (arr[i]==key)

        {

            printf("Elements is present at %d place.",i+1);

            flag=1;

            break;

        }

    }

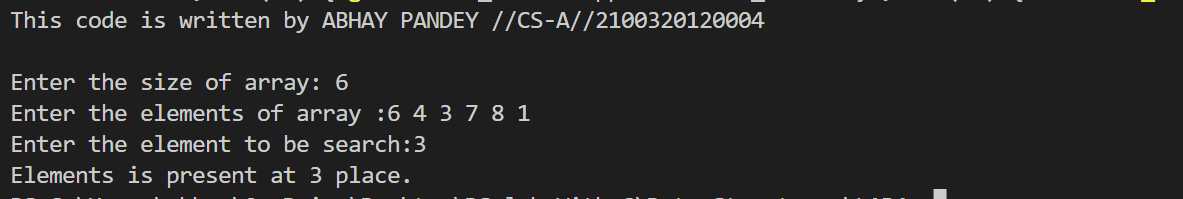
      if(flag==0)

      printf("Element is not present in array !!!");

      return 0 ;

 }

OUTPUT:



**PROGRAM 19 - Program for Matrix Addition**

**ALGORITHM: Matrixadd(A[][], B[][], M,N)**

**BEGIN:**C[M][N]

FOR i=1 TO M DO

FOR j=1 TO N DO

C[i][j]=A[i][j]+B[i][j]

RETURN C

**END;**

**Time Complexity: ϴ(N2)**

**Space Complexity:ϴ(N2)**

Source Code :

#include <stdio.h>

#include <math.h>

int main()

{ printf("This code is written by ABHAY PANDEY //CS-A//2100320120004 \n");

int m, n, o, p;

printf("Enter the row and column of first matrix : \n");

scanf("%d %d", &m, &n);

int a[m][n];

printf("Enter elements of first matrix : \n");

for (int i = 0; i < m; i++)

{

for (int j = 0; j < n; j++)

{

scanf("%d", &a[i][j]);

}

}

printf("Enter the row and column of second matrix : \n");

scanf("%d %d", &o, &p);

int b[o][p];

printf("Enter elements of second matrix : \n");

for (int i = 0; i < o; i++)

{

for (int j = 0; j < p; j++)

{

scanf("%d", &b[i][j]);

}

}

if (n == o)

{

printf("Addition of matrix is : \n");

for (int i = 0; i < m; i++)

{

for (int j = 0; j < n; j++)

{

printf("%d ", (a[i][j] + b[i][j]));

}

printf("\n");

}

}

return 0;

}

**Output:**

Text

Description automatically generated

**PROGRAM 20 - Program for Matrix Multiplication**

**ALGORITHM: Matrixmultiply(A[][], M,N, B[][], P,Q)**

**BEGIN:**

C[M][Q]

IF N!=P THEN

FOR i=1 TO M DO

FOR j=1 T0 Q DO

C[i][j]=0

FOR k=1 TO N DO

C[i][j]=C[i][j]+A[i][k]\*B[k][j]

RETURN C

**END;**

**Time Complexity: ϴ(N3)**

**Space Complexity:ϴ(N2)**

#include<stdio.h>

int main()

{

int n,m,p,q;

printf("This code is written by ABHAY PANDEY //CS-A//2100320120004 \n");

printf("Enter the rows and columns of matrix A and B-");

scanf("%d%d%d%d",&n,&m,&p,&q);

if(m==p){

int arr[n][m];

int brr[m][q];

int ans[n][q];

printf("Enter the elements of matrix A-");

for(int i=0;i<n;i++){

for(int j=0;j<m;j++)

scanf("%d",&arr[i][j]);

}

printf("Enter the elements of matrix B-");

for(int i=0;i<m;i++){

for(int j=0;j<q;j++)

scanf("%d",&brr[i][j]);

}

for(int i=0;i<n;i++){

for(int j=0;j<q;j++)

ans[i][j]=0;

}

for(int i=0;i<n;i++){

for(int j=0;j<q;j++){

for(int k=0;k<m;k++)

ans[i][j]+=arr[i][k]\*brr[k][j];

}

}

printf("Multiplication of matrix A and B is-");

for(int i=0;i<n;i++){

for(int j=0;j<q;j++)

printf("%d ",ans[i][j]);

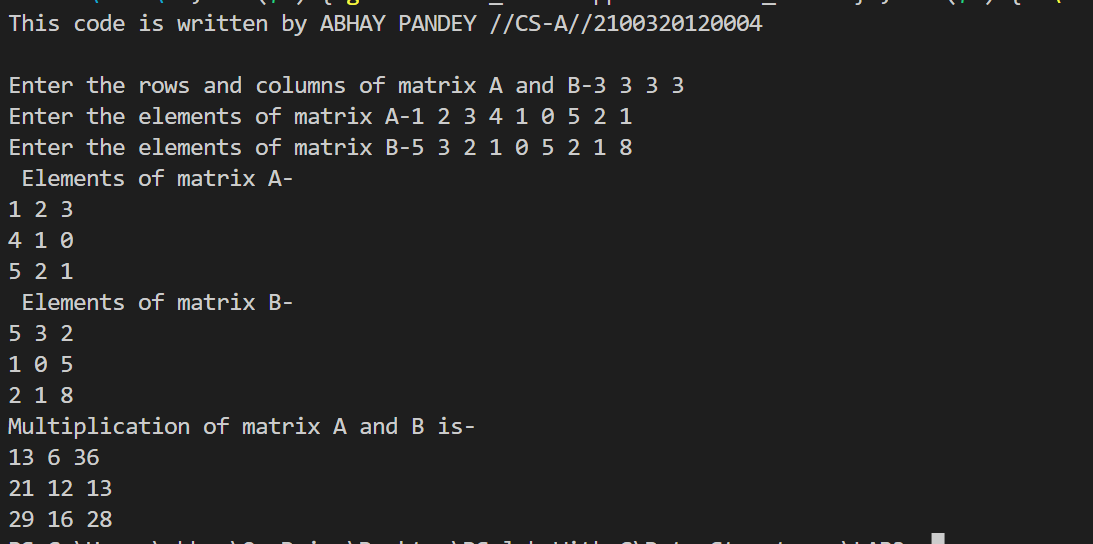
printf("\n");

}}

return 0 ;

}

**Output:**

**PROGRAM 21 - Program for Transpose of matrix using second matrix**

**ALGORITHM: Matrix\_transpose (A[][], M,N)**

**BEGIN:**

B[N][M]

FOR I =1 TO M DO

FOR j=1 TO N DO

B[j][i]=A[i][j]

RETURN B

**END;**

**Time Complexity: ϴ(N2)**

**Space Complexity:ϴ(N2**

#include <stdio.h>

#include <math.h>

int main()

{ printf("This code is written by ABHAY PANDEY //CS-A//2100320120004 \n");

int n, m;

printf("Enter the row and column of matrix : \n");

scanf("%d %d", &m, &n);

int a[n][m];

int t[m][n];

printf("Enter the elements of matrix : \n");

for (int i = 0; i < n; i++)

{

for (int j = 0; j < m; j++)

{

scanf("%d", &a[i][j]);

}

}

printf("The input matrix is \n");

for (int i = 0; i < n; i++)

{

for (int j = 0; j < m; j++)

{

printf("%d ", a[i][j]);

}

printf("\n");

}

for (int i = 0; i < n; i++)

{

for (int j = 0; j < m; j++)

{

t[i][j] = a[j][i];

}

}

printf("Transpose of matrix is : \n");

for (int i = 0; i < m; i++)

{

for (int j = 0; j < n; j++)

{

printf("%d ", t[i][j]);

}

printf("\n");

}

return 0;

}

Output:

Text, application

Description automatically generated**PROGRAM 6 - Program for Index** **Sequential Search**

**ALGORITHM: INDsearch(data[N],KEY,index[M][2])**

**BEGIN:**

FOR i=0 TO M-1 DO

IF KEY==index[i][1] THEN

RETURN index[i][0]

ELSE

IF KEY <index[i][1] THEN

high=index[i][0]-1

Low =index[i-1][0]+1

BREAK

FOR i=low TO high DO

IF KEY ==data[i] THEN

RETURN i

RETURN -1

**END;**

**Worst Case Time Complexity: O(N/K+K)**

**Best Case Time Complexity: 𝞨(1)**

**Space Complexity: ϴ(1)**

#include<stdio.h>

int index\_search(int arr[],int n,int key)

{

    int m=0,start,end,flag=0;

    int index[n/3],indexEle[n/3];

    for(int i=0;i<n;i+3)

{

        indexEle[m]=arr[i];

        index[m]=i;

        m++;

    }

    if(key<indexEle[0])

        return -1;

    else

{

        for(int i=1;i<m;i++)

{

            if(key<indexEle[i])

{

                start=index[i=1];

                end=index[i];

                flag=1;

                break;

            }

            if(flag==0)

{

                start=index[i-1];

                end=n-1;

            }

        }

    }

    for(int i=start;i<end;i++)

{

        if(arr[i]==key)

        return i;

    }

    return -1;

}

int main()

{ printf("This code is written by ABHAY PANDEY //CS-A//2100320120004 \n");

    int n;

    printf("Enter the size of array:");

    scanf("%d",&n);

    int arr[n];

    printf("Enter the elements of array:");

    for(int i=0;i<n;i++)

    scanf("%d",&arr[i]);

    int key;

    printf("Enter the element to be search:");

    scanf("%d",&key);

    int ans=index\_search(arr,n,key);

    if(ans==-1)

    printf("Element not found!!");

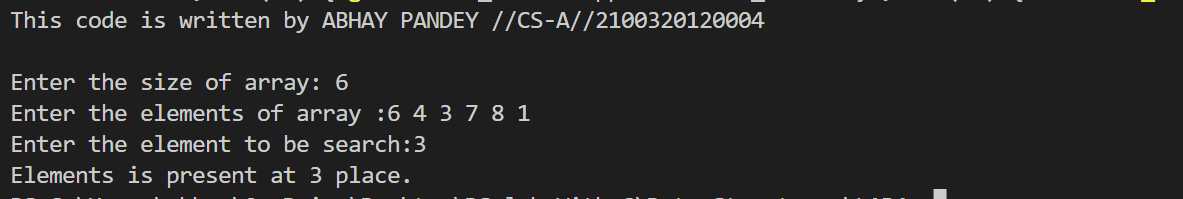
    else

    printf("Element is present at %d place.", ans+1);

    return 0 ;

}

**Output:**

****

**PROGRAM 18 - Program for Radix Sort**

**ALGORITHM: RadixSort(A[],N,d)**

**BEGIN:**

FOR i=1 TO d DO

Apply counting Sort on A[] at radix i

**END;**

**Time Complexity: ϴ(N)**

**Space Complexity:ϴ(N)**

#include <stdio.h>

int getMax(int a[], int n) {

int max = a[0];

for(int i = 1; i<n; i++) {

if(a[i] > max)

max = a[i];

}

return max;

}

void countingSort(int a[], int n, int place)

{

int output[n + 1];

int count[10] = {0};

for (int i = 0; i < n; i++)

count[(a[i] / place) % 10]++;

for (int i = 1; i < 10; i++)

count[i] += count[i - 1];

for (int i = n - 1; i >= 0; i--) {

count[(a[i] / place) % 10]--;

}

for (int i = 0; i < n; i++)

a[i] = output[i];

}

void radixsort(int a[], int n) {

int max = getMax(a, n);

for (int place = 1; max / place > 0; place \*= 10)

countingSort(a, n, place);

}

void printArray(int a[], int n) {

printf(“Sorted array”);

for (int i = 0; i < n; ++i) {

printf("%d ", a[i]);

}

printf("\n");

}

int main() {

int a[] = {181, 289, 390, 121, 145, 736, 514, 888, 122};

int n = sizeof(a) / sizeof(a[0]);

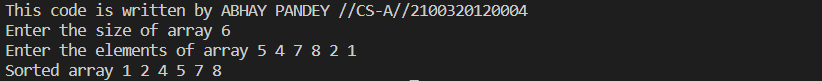
printf("This code is written by ABHAY PANDEY //CS-A//2100320120004 \n");

radixsort(a, n);

printArray(a, n);

}

**Output:**

**PROGRAM 17 - Program for Counting Sort**

**ALGORITHM: CountingSort(A[],k,n)**

**BEGIN:**

FOR i = 0 TO k DO

c[i] = 0

FOR j = 0 TO n DO

c[A[j]] = c[A[j]] + 1

FOR i = 1 TO k DO

c[i] = c[i] + c[i-1]

FOR j = n-1 TO 0 STEP-1 DO

B[ c[A[j]]-1 ] = A[j]

c[A[j]] = c[A[j]] - 1

RETURN B

**END;**

**Time Complexity: Omega(N)**

**Space Complexity:ϴ(N)**

#include <stdio.h>

void countingSort(int array[], int size) {

int output[10];

int max = array[0];

for (int i = 1; i < size; i++) {

if (array[i] > max)

max = array[i];

}

int count[10];

for (int i = 0; i <= max; ++i) {

count[i] = 0;

}

for (int i = 0; i < size; i++) {

count[array[i]]++;

}

for (int i = 1; i <= max; i++) {

count[i] += count[i - 1];

}

for (int i = size - 1; i >= 0; i--) {

output[count[array[i]] - 1] = array[i];

count[array[i]]--;

}

for (int i = 0; i < size; i++) {

array[i] = output[i];

}

}

void printArray(int array[], int size) {

printf("Sorted array ");

for (int i = 0; i < size; ++i) {

printf("%d ", array[i]);

}

printf("\n");

}

int main() {

int n;

printf("This code is written by ABHAY PANDEY //CS-A//2100320120004 \n");

printf("Enter the size of array ");

scanf("%d",&n);

int array[n];

printf("Enter the elements of array ");

for (int i = 0; i < n; i++)

{

scanf("%d",&array[i]);

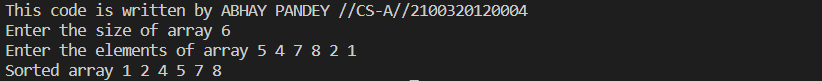
}

countingSort(array, n);

printArray(array, n);

}

Output:



**PROGRAM 7B - Program For Selection sort**

**ALGORITHM: SelectionSort(A[], N)**

**BEGIN:**

FOR i=1 TO N-1 DO

min=i

FOR j=i+1 TO N DO

IF A[j]<A[min] THEN

min=j

Exchange(A[min], A[i])

**END;**

**Time Complexity: ϴ(N2)**

**Space Complexity:ϴ(1)**

#include<stdio.h>

int main()

{

int n;

printf("This code is written by ABHAY PANDEY //CS-A//2100320120004 \n");

printf("Enter the size of array-");

scanf("%d",&n);

int arr[n];

printf("Enter the array:");

for (int i=0;i<n;i++){

scanf("%d",&arr[i]);

}

for (int i=0;i<n-1;i++)

{

for(int j=i+1;j<n;j++)

{ if(arr[j]<arr[i])

{

int temp=arr[j];

arr[j]=arr[i];

arr[i]=temp;

}

}

}

printf("Sorted array is :");

for(int i=0;i<n;i++)

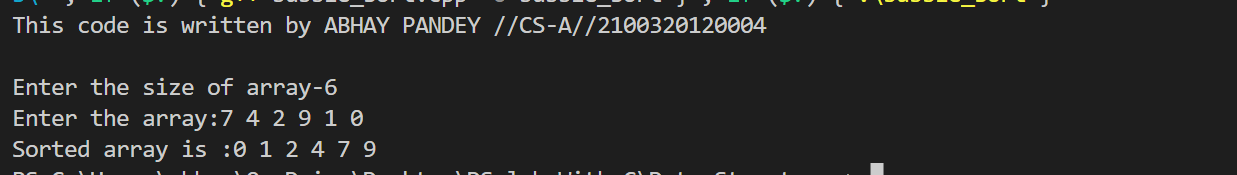
{

printf("%d ",arr[i]);

}

return 0 ;}

**Output:**



**PROGRAM 9 - Program for Quick sort**

**ALGORITHM: QuickSort(A[],low,high)**

**BEGIN:**

IF low<high THEN

j=Partition(A[],low,high)

QuickSort(A[],low,j-1)

QuickSort(A[],j+1,high)

**END;**

**ALGORITHM: Partition(A[],low,high)**

**BEGIN:**

i=low, j=high+1,pivot=A[low]

DO

DO

i=i+1

WHILE(A[i]<pivot)

DO

J=j-1

WHILE(A[j]>pivot)

IF i<j THEN

Exchange(A[i],A[j])

WHILE(i<j)

Exchange(A[j],A[low])

RETURN j

**END;**

**Worst Case Time Complexity:O(N2)**

**Best Case Time Complexity:** 𝜴**(Nlog**₂**N)**

**Space Complexity:** 𝚹**(log**₂**N)**

#include<stdio.h>

void swap(int arr[],int i,int j){

int temp=arr[i];

arr[i]=arr[j];

arr[j]=temp;

}

int partition(int arr[],int l,int r){

int pivot= arr[r];

int i=l-1;

for(int j=l;j<r;j++){

if(arr[j]<pivot)

{

i++;

swap(arr,i,j);

}

}

swap(arr,i+1,r);

return i+1;

}

void quickSort(int arr[],int l,int r){

if(l<r){

int pi=partition(arr,l,r);

quickSort(arr,l,pi-1);

quickSort(arr,pi+1,r);

}

}

int main()

{

int n;

printf("Enter size of array:");

scanf("%d",&n);

int arr[n];

printf("Enter array elements:");

for(int i=0;i<n;i++)

scanf("%d",&arr[i]);

quickSort(arr,0,n-1);

printf("Sorted array is: ");

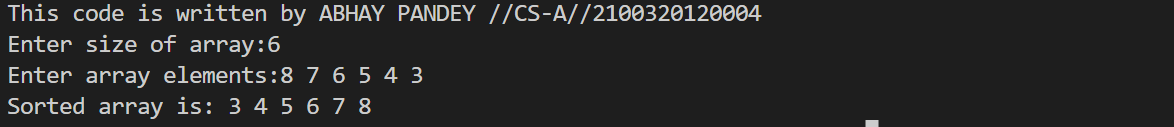
for(int i=0;i<n;i++)

printf("%d ",arr[i]);

return 0 ;

}

**Output:**

**PROGRAM 10 - Program for Merge sort**

**ALGORITHM: MergeSort(A[],low,high)**

**BEGIN:**

IF low<high DO

Mid=(low+high)/2

MergeSort(A[],low,mid)

MergeSort(A[],mid+1, high)

Merge(A, low,mid,high)

**END;**

**ALGORITHM: Merge(A[], low,mid,high)**

**BEGIN:**

i=low,j=mid+1,k=high

WHILE i<=mid AND j<=high DO

IF A[i]<A[j] THEN

C[k]=A[i]

i=i+1

k=k+1

ELSE

C[k]=A[j]

j=j+1

k=k+1

WHILE i<=mid DO

C[k]=A[i]

i=i+1

k=k+1

WHILE j<=high DO

C[k]=A[j]

J=j+1

k=k+1

FOR i=low TO high DO

A[i]=C[i]

**END;**

**Time Complexity: O(Nlog**₂**N)**

**Space Complexity:** 𝚹**(N)**

#include<stdio.h>

void merge (int arr[],int l,int mid,int r)

{

int n1=mid-l+1;

int n2=r-mid;

int a[n1];

int b[n2];

for (int i=0;i<n1;i++)

a[i]=arr[l+i];

for (int i=0;i<n2;i++)

b[i]=arr[mid+1+i];

int i=0;

int j=0;

int k=l;

while(i<n1 && j<n2)

{

if(a[i]<b[j])

{

arr[k]=a[i];

k++;

i++;

}

else

{

arr[k]=b[j];

k++;

j++;

}

}

while(i<n1){

arr[k]=a[i];

k++;

i++;

}

while(j<n2)

{

arr[k]=b[j];

k++;

j++;

}

}

void mergeSort(int arr[],int l,int r)

{

if(l<r){

int mid=(l+r)/2;

mergeSort(arr,l,mid);

mergeSort(arr,mid+1,r);

merge(arr,l,mid,r);

}

}

int main()

{ int n;

printf("Enter size of array:");

scanf("%d",&n);

int arr[n];

printf("Enter array elements:");

for(int i=0;i<n;i++)

{

scanf("%d",&arr[i]);

}

mergeSort(arr,0,n-1);//l=0 r=n-1

printf(“Sorted array is:”);

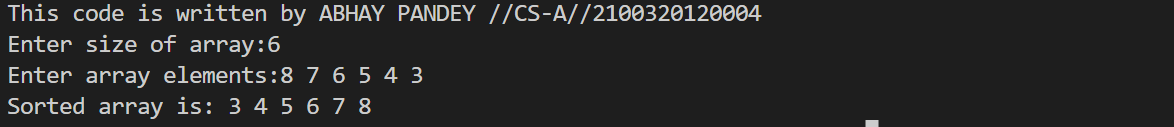
for(int i=0;i<n;i++)

printf("%d ",arr[i]);

return 0 ;

}

**Output:**



**PROGRAM 7C - Program for Insertion sort**

**ALGORITHM: InsertionSort(A[], N)**

**BEGIN:**

FOR i=2 TO N DO

key=A[i]

j=i-1

WHILE j>=1 AND A[j]>key DO

A[j+1]=A[j]

j=j-1

A[j+1]=key

**END;**

**Worst Case Time Complexity:O(N2)**

**Best Case Time Complexity: Omega(N)**

**Space Complexity:ϴ(1)**

#include<stdio.h>

int main()

{

int n;

printf("Enter size of array:");

scanf("%d",&n);

int arr[n];

printf("Enter array elements:");

for(int i=0;i<n;i++)

{

scanf("%d",&arr[i]);

}

for(int i=1;i<n;i++)

{

int current=arr[i];

int j=i-1;

while(arr[j]>current&&j>=0)

{

arr[j+1]=arr[j];

j--;

}

arr[j+1]=current;

}

printf("Sorted array is:");

for (int i=0;i<n;i++)

{

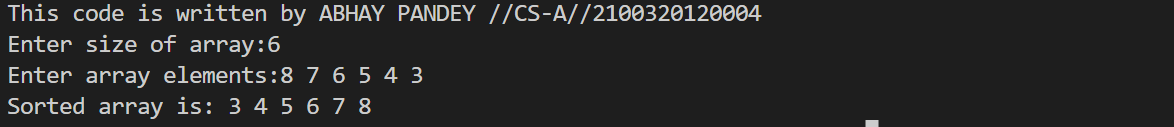
printf("%d ",arr[i]);

}

return 0 ;

}

**Output:**



**PROGRAM 7A - Program for Bubble sort**

**ALGORITHM: BubbleSort(A[], N)**

**BEGIN:**

FOR i=1 TO N-1 DO

FOR j=1 TO N-i DO

IF A[j]>A[j+1]

k=A[j]

A[j]=A[j+1]

A[j+1]=k

**END;**

**Worst Case Time Complexity:O(N2)**

**Best Case Time Complexity: Omega(N)**

**Space Complexity:ϴ(1)**

#include<stdio.h>

int main()

{

int n;

printf("This code is written by ABHAY PANDEY //CS-A//2100320120004 \n");

printf("Enter the size of array-");

scanf("%d",&n);

int arr[n];

printf("Enter the array:");

for(int i=0;i<n;i++){

scanf("%d",&arr[i]);

}

int count=1;

while(count<n){

for(int i=0;i<n-count;i++){

if(arr[i]>arr[i+1])

{int temp=arr[i];

arr[i]=arr[i+1];

arr[i+1]=temp;}

}

count++;

}

printf("Sorted array is :");

for(int i=0;i<n;i++){

printf("%d ",arr[i]);

}

return 0 ;

}

**Output:**

