C Programming Assignment

NAME-AKSHAT SHAH

ROLL NO-21BCP322

DVISION-3

GROUP-10

COMPUTER ENGINEERING

Date-30 December, 2021

Understanding the importance of loops in c language.

Here are the problems which explain us the concept of loops:

1. Write a program in C to display the first 10 natural numbers.

Source code:

```
#include <stdio.h>
  int main()
{
    int a;

    for (int num=1;num<=10;num++)
{
    printf("%d",num);
}

    return 0;
}
```

Expected output:

```
1 2 3 4 5 6 7 8 9 10
```

2. Write a C program to find the sum of first 10 natural numbers.

Source code:

```c
#include <stdio.h>
  int main()
{
        int  j, sum = 0;


        printf("The first 10 natural number is :\n");


        for (j = 1; j <= 10; j++)
 {
        sum = sum + j;
        printf("%d ",j);
  }
         printf("\nThe Sum is : %d\n", sum);


          return 0;
}
```

Expected output:

```
The first 10 natural number is :
1 2 3 4 5 6 7 8 9 10
The Sum is : 55
```

3. Write a program in C to display n terms of natural number and their sum.

Source code:

```
#include <stdio.h>
  int main()
{
        int n, i, sum = 0;
        printf("Enter a positive integer: ");
        scanf("%d", &n);
         i = 1;
         while (i <= n)
 {
         sum += i;
           ++i;
    }

   printf("Sum of the  terms upto that:   = %d", sum);
   return 0;
    }
```

Input:

```
Enter a positive integer: 32
```

Output:

```
Sum of the  terms upto that :528
```

4.Write a program in C to read 10 numbers from keyboard and find their sum and average.

Source code:

```
#include <stdio.h>
void main()
{
        int i,num,sum=0;
            float avg;
            printf("Input the 10 numbers : \n");
        for (i=1;i<=10;i++)
{
         printf("No.-%d :",i);

                 scanf("%d",&num);
                 sum +=num;
}
            avg=sum/10;
  printf("The sum of 10 nos is : %d\nThe   Average is :
%f\n",sum,avg);
}
```

```
Input:

Input the 10 numbers :

No.-1:23

No.-2:24

No.-3:54

No.-4:32

No.-5:12

No.-6:76

No.-7:65

No.-8:94

No.-9:48
```

No.-10:35

Output:

```
The sum of 10 nos is : 463
The   Average is:46.2999
```

5.Write a program in C to display the cube of the number upto given an integer.

Source code:

```
#include <stdio.h>
void main()
{
         int i,cube;
        printf("Input number of terms : ");
        scanf("%d", &cube);
        for(i=1;i<=cube;i++)
```

```
 {
        printf("Number is : %d and cube of the %d is :%d
\n",i,i, (i*i*i));
}


}
```

Input:

```
Input number of terms:6
```

Output:

```
Number is : 1 and cube of the 1 is :1
Number is : 2 and cube of the 1 is :8
Number is : 3 and cube of the 1 is :27
Number is : 4 and cube of the 1 is :64
Number is : 5 and cube of the 1 is :125
Number is : 6 and cube of the 1 is : 216
```

6. Write a program in C to display the multiplication table of a given integer.

Source code:

```
#include <stdio.h>
int main()
{
 int num, i;
 printf("Enter an integer: ");
 scanf("%d", &num);
 for (i = 1; i <= 10; ++i)
{
   printf("%d * %d = %d \n", num, i, num * i);
}
   return 0;
}
```

Input:

```
Enter an integer: 7
```

Output:

```
7*1=7

7*2=14

7*3=21

7*4=28

7*5=35

7*6=42

7*7=49

7*8=56

7*9=63

7*10=70
```

7.Write a program in C to display the multiplication table vertically from 1 to n.

Source code:

```
#include <stdio.h>
  void main()
{
        int j,i,num;
        printf("Input upto the table number starting
        from 1 : ");
         scanf("%d",&num);
         printf("Multiplication table from 1 to %d \n",num);
        for(i=1;i<=10;i++)
 {
        for(j=1;j<=num;j++)
  {
         if (j<=num-1)
```

```
          printf("%dx%d = %d, ",j,i,i*j);
          else
           printf("%dx%d = %d\n",j,i,i*j);


  }
           printf("\n");
   }
   }
```

Input:

```
Input upto the table number starting from 1 : 10
```

Output:

```
Multiplication table from 1 to 10
1x1 = 1, 2x1 = 2, 3x1 = 3, 4x1 = 4, 5x1 = 5, 6x1 = 6, 7x1 =
7, 8x1 = 8, 9x1 = 9, 10x1 = 10
```

1x2 = 2, 2x2 = 4, 3x2 = 6, 4x2 = 8, 5x2 = 10, 6x2 = 12, 7x2 = 14, 8x2 = 16, 9x2 = 18, 10x2 = 20

1x3 = 3, 2x3 = 6, 3x3 = 9, 4x3 = 12, 5x3 = 15, 6x3 = 18, 7x3 = 21, 8x3 = 24, 9x3 = 27, 10x3 = 30

1x4 = 4, 2x4 = 8, 3x4 = 12, 4x4 = 16, 5x4 = 20, 6x4 = 24, 7x4 = 28, 8x4 = 32, 9x4 = 36, 10x4 = 40

1x5 = 5, 2x5 = 10, 3x5 = 15, 4x5 = 20, 5x5 = 25, 6x5 = 30, 7x5 = 35, 8x5 = 40, 9x5 = 45, 10x5 = 50

1x6 = 6, 2x6 = 12, 3x6 = 18, 4x6 = 24, 5x6 = 30, 6x6 = 36, 7x6 = 42, 8x6 = 48, 9x6 = 54, 10x6 = 60

1x7 = 7, 2x7 = 14, 3x7 = 21, 4x7 = 28, 5x7 = 35, 6x7 = 42, 7x7 = 49, 8x7 = 56, 9x7 = 63, 10x7 = 70

```
1x8 = 8, 2x8 = 16, 3x8 = 24, 4x8 = 32, 5x8 = 40, 6x8 = 48,
7x8 = 56, 8x8 = 64, 9x8 = 72, 10x8 = 80


1x9 = 9, 2x9 = 18, 3x9 = 27, 4x9 = 36, 5x9 = 45, 6x9 = 54,
7x9 = 63, 8x9 = 72, 9x9 = 81, 10x9 = 90


1x10 = 10, 2x10 = 20, 3x10 = 30, 4x10 = 40, 5x10 = 50,
6x10 = 60, 7x10 = 70, 8x10 = 80, 9x10 = 90, 10x10 = 100
```

8.Write a program in C to display the n terms of odd natural number and their sum.

Source code :

```c
#include <stdio.h>
void main()
{
  int i,num,sum=0;
```

```
   printf("Input number of terms : ");
   scanf("%d",&num);
   printf("\nThe odd numbers are :");
   for(i=1;i<=num;i++)
   {
     printf("%d ",2*i-1);
     sum+=2*i-1;
   }
   printf("\nThe Sum of odd Natural Number upto %d
terms : %d \n",num,sum);
}
```

Input:

```
Input number of terms : 6
```

Output:

```
The odd numbers are :1 3 5 7 9 11
The Sum of odd Natural Number upto 6 terms : 36
```

9. Write a program in C to display the pattern like right angle triangle using an asterisk.

Source code:

```
#include <stdio.h>
int main()
{

 for(int i=0;i<=4;i++)
{
 for(int j=0;j<=4;j++)
{
 if (i>j)
{
 printf("*");
}
 else{
 printf(" ");
}
}
 printf("\n");
}
 return 0;
}
```

Output:

```
*
**
***
****
```

10.Write a program in C to display the pattern like right angle triangle with a number.

Source code:

```
#include <stdio.h>

void main()

{

   int i,j,rows;

   printf("Input number of rows : ");

   scanf("%d",&rows);

   for(i=1;i<=rows;i++)

{

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

        printf("%d",j);

        printf("\n");

}

}
```

Input:

Input number of rows : 6

Output:

```
1

12

123

1234

12345

123456
```

11. Write a program in C to make such a pattern like right angle triangle with a number which will repeat a number in a row.

Source code:

```
#include <stdio.h>

int main()

{
 for(int i=0;i<=4;i++)
{
 for(int j=0;j<=4;j++)
{
 if (i>j)
{
 printf("%d",i);
}
 else
 {
 printf("\n");
}
```

Output:

```
1
22
333
```

```
4444
```

12. Write a program in C to make such a pattern like right angle triangle with number increased by 1.

Source code:

```
#include <stdio.h>
int main() {
 int n=1;
for(int i=0;i<=4;i++)
 {
 for(int j=0;j<=4;j++)
 {
 if (i>j)
 {
 printf("%d ",n++);
 }
 else
{
 printf(" ");
 }
 }
 printf("\n");
 }
 return 0;
}
```

Output:

```
1
2 3
4 5 6
7 8 9 10
```

13.Write a program in C to make such a pattern like a pyramid with numbers increased by 1.

Source code:

```
#include <stdio.h>
void main()
{
   int i,j,spc,rows,k,t=1;
   printf("Input number of rows : ");
   scanf("%d",&rows);
   spc=rows+4-1;
   for(i=1;i<=rows;i++)
{
    for(k=spc;k>=1;k--)
{
    printf(" ");


}

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

     printf("%d ",t++);

         printf("\n");

     spc--;

}

}
```

Output:

```
Input number of rows : 4

       1

      2 3

     4 5 6

    7 8 9 10
```

14.Write a program in C to make such a pattern like a pyramid with an asterisk.

Source code:

```
  #include <stdio.h>
void main()
{
   int i,j,samp,rows,k;
   printf("Input number of rows : ");
   scanf("%d",&rows);
   samp=rows+4-1;
   for(i=1;i<=rows;i++)
   {
         for(k=samp;k>=1;k--)
            {
              printf(" ");
            }
```

```
          for(j=1;j<=i;j++)
          printf("* ");
        printf("\n");
    samp--;
   }
}
```

Input:

Input number of rows : 4

Output:

```
      *
     * *
    * * *
   * * * *
```

15. Write a C program to calculate the factorial of a given number.

Source code:

```
#include <stdio.h>
void main()
{
  int i,fct=1,num;

  printf("Input the number : ");
  scanf("%d",&num);

  for(i=1;i<=num;i++)
      fct=fct*i;

  printf("The Factorial of %d is: %d\n",num,fct);
}
```

Input:

```
Input the number : 23
Output:
The Factorial of 23 is: 862453760
```

16. Write a program in C to display the n terms of even natural number and their sum.

Source code:

```
#include <stdio.h>

void main()
{
  int i,num,sum=0;

  printf("Input number of terms : ");
  scanf("%d",&num);
```

```
   printf("\nThe even numbers are :");
   for(i=1;i<=num;i++)
   {
     printf("%d ",2*i);
     sum+=2*i;
   }
   printf("\nThe Sum of even Natural Number upto %d
terms : %d \n",num,sum);
}
```

Input:

```
Input number of terms : 7
```

Output:

```
The even numbers are :2 4 6 8 10 12 14
The Sum of even Natural Number upto 7 terms : 56
```

17. Write a program in C to make such a pattern like a pyramid with a number which will repeat the number in the same row.

Source code:

```
#include <stdio.h>

void main()
{
   int i,j,samp,rows,k;
   printf("Input number of rows : ");
   scanf("%d",&rows);
   samp=rows+4-1;
   for(i=1;i<=rows;i++)
   {
         for(k=samp;k>=1;k--)
```

```
            {
             printf(" ");
            }

           for(j=1;j<=i;j++)
           printf("%d ",i);
        printf("\n");
    samp--;
   }
}
```

Input:

Input number of rows : 5

Output:

```
        1
```

```
  2 2
 3 3 3
4 4 4 4
5 5 5 5 5
```

18. Write a program in C to find the sum of the series [1-X^2/2!+X^4/4!- ....].

Source code:

```
#include <stdio.h>
void main()
{
        float z,sum,t,d;
        int i,num;
        printf("Input the Value of z :");
        scanf("%f",&z);
        printf("Input the number of terms : ");

        scanf("%d",&num);

        sum =1; t = 1;

        for (i=1;i<num;i++)

        {

          d = (2*i)*(2*i-1);

          t = -t*z*z/d;

          sum =sum+ t;

        }

        printf("\nthe sum = %f\nNumber of terms =
%d\nvalue of z = %f\n",sum,num,z);

}
```

Input:

```
Input the Value of z :4

Input the number of terms : 4
```

Output:

```
the sum = -2.022222
Number of terms = 4
value of z = 4.000000
```

19. Write a program in C to display the n terms of harmonic series and their sum.

1 + 1/2 + 1/3 + 1/4 + 1/5 ... 1/n terms.

Source code:

```
#include <stdio.h>
void main()
{
   int i,num;
   float k=0.0;
```

```
printf("Input the number of terms : ");

scanf("%d",&num);

printf("\n\n");

for(i=1;i<=num;i++)

{

    if(i<num)

{

  printf("1/%d + ",i);

  k+=1/(float)i;

    }

  if(i==num)

  {

  printf("1/%d ",i);

  k+=1/(float)i;

  }

  }
```

```
	printf("\nSum of Series upto %d terms : %f
\n",num,k);
}
```

Input:

```
Input the number of terms : 5
```

Output:

```
1/1 + 1/2 + 1/3 + 1/4 + 1/5
Sum of Series upto 5 terms : 2.283334
```

20. Write a program in C to display the pattern like a pyramid using asterisk and each row contain an odd number of asterisks.

Source code:

21.Write a program in C to display the sum of the series [9 + 99 + 999 + 9999 ...].

Source code:

```
#include <stdio.h>
    void main()
{
    long int num,i,k=9;
        int sum =0;
        printf("Input the number or terms :");
        scanf("%ld",&num);
        for (i=1;i<=num;i++)
{
            sum +=k;
            printf("%ld ",k);
            k=k*10+9;
}
            printf("\nThe sum of the series = %d \n",sum);
}
```

Input:

```
Input the number or terms :6
```

Output:

```
9   99   999   9999   99999   999999
The sum of the series = 1111104
```

22. Write a program in C to print the Floyd's Triangle.

Source code:

```
void main()
{
   int i,j,num,p,q;
   printf("Input number of rows : ");
   scanf("%d",&num);
```

```
for(i=1;i<=num;i++)
{
  if(i%2==0)
  {
      p=1;q=0;


  }
  else
  {
      p=0;q=1;


  }


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


      if(j%2==0)
      printf("%d",p);
```

```
        else
        printf("%d",q);
     printf("\n");
   }
}
```

Input:

Input number of rows : 5

Output:

```
1
01
101
0101
10101
```

23. Write a program in C to display the sum of the series [1+x+x^2/2!+x^3/3!+....]

Source code:

```
#include <stdio.h>

void main()
{
        float x,sum,no_row;
        int i,num;
        printf("Input the value of x :");
        scanf("%f",&x);
        printf("Input number of terms : ");
        scanf("%d",&num);
        sum =1; no_row = 1;
        for (i=1;i<num;i++)
        {
          no_row = no_row*x/(float)i;
          sum =sum+ no_row;
        }
        printf("\nThe sum  is : %f\n",sum);
}
```

Input:

```
Input the value of x :4
Input number of terms : 3
```

output:

```
The sum  is : 13.000000
```

24. Write a program in C to find the sum of the series [x - x^3 + x^5 + ...].

Source code:

```
void main()
{
        int x,sum,ctr;
```

```
    int i,p,q,pp,qq;
    printf("Input the value of x :");
    scanf("%d",&x);
    printf("Input number of terms : ");
    scanf("%d",&q);
    sum =x; p=-1;
    printf("The values of the series: \n");
    printf("%d\n",x);
for (i = 1; i < q; i++)
{
    ctr = (2 * i + 1);
    pp = pow(x, ctr);
    qq = pp * p;
    printf("%d \n",qq);
    sum = sum + qq;
    p = p * (-1);
    }
```

```
        printf("\nThe sum = %d\n",sum);
}
```

Input:

```
Input the value of x :2
Input number of terms : 3
```

Output:

```
The values of the series:
2
-8
32
The sum = 26
```

25. Write a program in C to display the n terms of square natural number and their sum.

1 4 9 16 ... n Terms.

Source code:

```
#include <stdio.h>


void main()

{

   int i,num,sum=0;

   printf("Input the number of terms : ");

   scanf("%d",&num);

   printf("\nThe square natural upto %d terms are
:",num);

   for(i=1;i<=num;i++)

   {

     printf("%d  ",i*i);

     sum+=i*i;

   }

   printf("\nThe Sum of Square Natural Number upto %d
terms = %d \n",num,sum);
}
```

Input:

```
Input the number of terms : 3
```

Output:

```
The square natural upto 3 terms are :1  4  9
The Sum of Square Natural Number upto 3 terms = 14
```

26. Write a program in C to find the sum of the series 1 +11 + 111 + 1111 + .. n terms.

Source code:

```
#include <stdio.h>

void main()
{
```

```
int num,i;

long sum=0;

long int t=1;

printf("Input the number of terms : ");

scanf("%d",&num);

for(i=1;i<=num;i++)

{

   printf("%ld",t);

    if (i<num)

    {

        printf("+ ");


    }

   sum=sum+t;

   t=(t*10)+1;

}

printf("\nThe Sum is : %ld\n",sum);
```

```
}
```

Input:

```
Input the number of terms : 7
```

Output:

```
1+ 11+ 111+ 1111+ 11111+ 111111+ 1111111
The Sum is : 1234567
```

27. Write a c program to check whether a given number is a perfect number or not.

Source code:

```
 #include <stdio.h>

  void  main()
```

```
{
    int n,i,sum;
    int pn,px;

    printf("Input the  number : ");
     scanf("%d",&n);
    sum = 0;
    printf("The positive divisor  : ");
    for (i=1;i<n;i++)
{
     if(n%i==0)
{
     sum=sum+i;
    printf("%d  ",i);
}
}
    printf("\nThe sum of the divisor is : %d",sum);
    if(sum==n)
      printf("\nSo, the number is perfect.");
    else
      printf("\nSo, the number is not perfect.");
    printf("\n");
}
```

Input:

```
Input the  number : 45
```

output:

```
The positive divisor  : 1  3  5  9  15
The sum of the divisor is : 33
So, the number is not perfect.
```

28. Write a c program to find the perfect numbers within a given number of range.

Source code:

```
#include <stdio.h>

void  main()
{
  int n,i,sum;
  int mn,mx;
  printf("Input the starting range or number : ");
  scanf("%d",&mn);
  printf("Input the ending range of number : ");
  scanf("%d",&mx);
  printf("The Perfect numbers within the given range : ");
  for(n=mn;n<=mx;n++)
{
    i=1;
    sum = 0;
    while(i<n)
{
    if(n%i==0)
    sum=sum+i;
    i++;
}
    if(sum==n)
    printf("%d ",n);
}
    printf("\n");
}
```

Input:

```
Input the starting range or number : 1
Input the ending range of number : 100
```

Output:

```
The Perfect numbers within the given range : 6 , 28
```

29. Write a C program to check whether a given number is an armstrong number or not.

Source code:

```
#include <stdio.h>

void main()
{
   int num,r,sum=0,i;

   printf("Input  a number: ");
   scanf("%d",&num);

   for(i=num;num!=0;num=num/10){
       r=num % 10;
```

```
         sum=sum+(r*r*r);

}

    if(sum==i)

     printf("%d is an Armstrong number.\n",i);

    else


     printf("%d is not an Armstrong number.\n",i);


}
```

Input:

```
Input  a number: 4581
```

output:

```
4581 is not an Armstrong number.
```

30. Write a C program to find the Armstrong number for a given range of number.

Source code:

```
#include <stdio.h>


void main()
{
   int num,r,sum,temp;
   int stno,enno;


   printf("Input starting number of range: ");
   scanf("%d",&stno);


   printf("Input ending number of range : ");
   scanf("%d",&enno);
```

```
    printf("Armstrong numbers in given range are: ");


    for(num=stno;num<=enno;num++)
{
        temp=num;
        sum = 0;


        while(temp!=0)
{
            r=temp % 10;
            temp=temp/10;
            sum=sum+(r*r*r);
}
        if(sum==num)
            printf("%d ",num);
}
printf("\n");
```

```
}
```

```
Input:
Input starting number of range: 1
Input ending number of range : 1000
Output:
Armstrong numbers in given range are: 1 153 370 371
407
```

31. Write a program in C to display the pattern like a diamond.

```
 #include <stdio.h>


void main()
{
  int i,j,k;
  printf("Input number of rows (half of the diamond) :");
```

```
   scanf("%d",&k);
   for(i=0;i<=k;i++)
{
     for(j=1;j<=k-i;j++)
     printf(" ");
     for(j=1;j<=2*i-1;j++)
       printf("*");
     printf("\n");
}

   for(i=k-1;i>=1;i--)
{
     for(j=1;j<=k-i;j++)
     printf(" ");
     for(j=1;j<=2*i-1;j++)
     printf("*");
     printf("\n");
}

}
```

Input:

```
Input number of rows (half of the diamond) :5
```

Output:

```
    *
   ***
  *****
 *******
*********
 *******
  *****
   ***
    *
```

32.Write a C program to determine whether a given number is prime or not.

Source code:

```
#include <stdio.h>
void main()
{

   int num,i,ctr=0;
   printf("Input  a number: ");
   scanf("%d",&num);
   for(i=2;i<=num/2;i++)
{
      if(num % i==0)
{
```

```
         ctr++;
            break;
}
}
   if(ctr==0 && num!= 1)
        printf("%d is a prime number.\n",num);
   else
      printf("%d is not a prime number",num);
}
```

Input:

Input a number: 123439

output:

```
123439 is a prime number.
```

33. Write a C program to display Pascal's triangle.

Source code:

```
#include <stdio.h>
void main()
{
  int no_row,c=1,blk,i,j;
  printf("Input number of rows: ");
  scanf("%d",&no_row);
  for(i=0;i<no_row;i++)
{
    for(blk=1;blk<=no_row-i;blk++)
    printf("  ");
    for(j=0;j<=i;j++)
{
      if (j==0||i==0)
        c=1;
      else
       c=c*(i-j+1)/j;
      printf("% 4d",c);
}
    printf("\n");
}
}
```

Input:

Input number of rows: 6

Output:

```
        1
      1   1
    1   2   1
  1   3   3   1
1   4   6   4   1
1   5  10  10   5   1
```

34. Write a program in C to find the prime numbers within a range of numbers.

Source code:

```
#include <stdio.h>


void main()

{

    int num,i,ctr,stno,enno;


    printf("Input starting number of range: ");

    scanf("%d",&stno);


    printf("Input ending number of range : ");

    scanf("%d",&enno);

    printf("The prime numbers between %d and %d are : 
\n",stno,enno);
```

```
    for(num = stno;num<=enno;num++)
{
        ctr = 0;


        for(i=2;i<=num/2;i++)
    {
            if(num%i==0){
                ctr++;
                break;
     }
}
      if(ctr==0 && num!= 1)
            printf("%d ",num);
    }
printf("\n");
}
```

```
Input:

Input starting number of range: 32

Input ending number of range : 230


Output:

The prime numbers between 32 and 230 are :

37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107
109 113 127 131 137 139 149 151 157 163 167 173 179
181 191 193 197 199 211 223 227 229
```

35. Write a program in C to display the first n terms of Fibonacci series.

source code:

```c
#include <stdio.h>


void main()

{
   int prv=0,pre=1,trm,i,n;

   printf("Input number of terms to  display : ");

   scanf("%d",&n);

   printf("Here is the Fibonacci series upto  to %d terms :
\n",n);

   printf("% 5d % 5d", prv,pre);


  for(i=3;i<=n;i++)

   {

     trm=prv+pre;

     printf("% 5d",trm);

     prv=pre;

     pre=trm;

   }

   printf("\n");

}
```

Input:

Input number of terms to display : 7

Output:

```
Here is the Fibonacci series upto  to 7 terms :
   0    1    1    2    3    5    8
```

36. Write a program in C to display the such a pattern for n number of rows using a number which will start with the number 1 and the first and a last number of each row will be 1.

Source code:

```
#include <stdio.h>


void main()
{
   int i,j,num;
   printf("Input number of rows : ");
   scanf("%d",&num);
   for(i=0;i<=num;i++)
   {

     for(j=1;j<=num-i;j++)
        printf(" ");
     for(j=1;j<=i;j++)
       printf("%d",j);

       for(j=i-1;j>=1;j--)
          printf("%d",j);

     printf("\n");
   }
}
```

Input:

Input number of rows : 5

Output:

```
    1
   121
  12321
 1234321
123454321
```

37. Write a program in C to display the number in reverse order.

Source code:

```
#include <stdio.h>


void main()
{
    int num,r,sum=0,t;
    printf("Input a number: ");
    scanf("%d",&num);

    for(t=num;num!=0;num=num/10)
{
        r=num % 10;
        sum=sum*10+r;
}
    printf("The number in reverse order is : %d \n",sum);
}
```

Input:

```
Input a number: 453234
```

Output:

```
The number in reverse order is : 432354
```

38. Write a program in C to check whether a number is a palindrome or not.

Source code:

```
#include <stdio.h>


void main()

{

   int num,r,sum=0,t;


   printf("Input a number: ");

   scanf("%d",&num);


   for(t=num;num!=0;num=num/10)

{

      r=num % 10;

      sum=sum*10+r;

}
```

```
    if(t==sum)
        printf("%d is a palindrome number.\n",t);
    else
         printf("%d is not a palindrome number.\n",t);


}
```

Input:

```
Input a number: 1331
```

Output:

```
1331 is a palindrome number.
```

39. Write a program in C to find the number and sum of all integer between 100 and 200 which are divisible by 9.

Source code:

```
#include <stdio.h>
```

```
void main()
{
   int i, sum=0;
   printf("Numbers between 100 and 200, divisible by 9 :
\n");
   for(i=101;i<200;i++)
{
     if(i%9==0)
{
       printf("% 5d",i);
       sum+=i;
}
}
   printf("\n\nThe sum : %d \n",sum);
}
```

output:

```
 Numbers between 100 and 200, divisible by 9 :
  108  117  126  135  144  153  162  171  180  189  198
The sum : 1683
```

40. Write a C Program to display the pattern like pyramid using the alphabet.

Source code:

```
#include <stdio.h>


void main()
{
   int i, j;
   char alph = 'A';
   int n,blk;
   int ctr = 1;
   printf("Input the number of Letters (less than 26) in the
Pyramid : ");
   scanf("%d", &n);


   for (i = 1; i <= n; i++)
{
        for(blk=1;blk<=n-i;blk++)


        printf("  ");
    for (j = 0; j <= (ctr / 2); j++)
{
     printf("%c ", alph++);
}
     alph = alph - 2;


     for (j = 0; j < (ctr / 2); j++)
{
     printf("%c ", alph--);
}
      ctr = ctr + 2;
      alph = 'A';
      printf("\n");
}
}
```

Input:

```
Input the number of Letters (less than 26) in the Pyramid
: 5
```

Output:

```
         A
      A B A
```

```
    A B C B A
  A B C D C B A
A B C D E D C B A
```

41. Write a program in C to convert a decimal number into binary without using an array.

Source code:

```
#include <stdio.h>
#include <stdlib.h>
  char *decimal_to_binary(int);
  char *decimal_to_binary(int dn)
{
  int i, j, k;
  char *ptr;
  k = 0;
  ptr = (char*)malloc(32+1);
  for (i = 31 ; i >= 0 ; i--)
{
    j = dn >> i;
    if (j & 1)
      *(ptr+k) = 1 + '0';
    else
      *(ptr+k) = 0 + '0';
    k++;
}
  *(ptr+k) = '\0';
  return  ptr;
}
int main()
{
  int dn;
  char *ptr;
  printf("Input a decimal number: ");
  scanf("%d", &dn);
  ptr = decimal_to_binary(dn);
  printf("Binary number equivalent to said decimal number is: %s", ptr);
  free(ptr);
  return 0;
}
```

Input:

```
Input a decimal number: 25
```

Output:

```
Binary number equivalent to said decimal number is: 00000000000000000000000000011001
```

42. Write a program in C to convert a binary number into a decimal number without using array, function and while loop.

Source code:

```
#include <stdio.h>


void main()
{      int k, n,p=1;
       int dec=0,i=1,j,d;


     printf("\n\n  Convert Binary to Decimal:\n ");
     printf("-------------------------\n");


       printf("Input a binary number :");
       scanf("%d",&n);
```

```
        k=n;
        for (j=n;j>0;j=j/10)
{
          d = j % 10;
            if(i==1)
            p=p*1;
            else
            p=p*2;

                dec=dec+(d*p);
                i++;
}
        printf("\nThe Binary Number : %d\nThe equivalent
Decimal  Number : %d \n\n",k,dec);
}
```

Input:

```
  Convert Binary to Decimal:

 -------------------------

Input a binary number :101000111


Output:

The Binary Number : 101000111

The equivalent Decimal  Number : 327
```

43. Write a C program to find HCF (Highest Common Factor) of two numbers.

Source code:

```
#include <stdio.h>


void main()


{

   int i, num1, num2, j, hcf=1;
```

```
    printf("\n\n  HCF of two numbers:\n ");
    printf("----------------------\n");



    printf("Input 1st number for HCF: ");
    scanf("%d", &num1);
    printf("Input 2nd number for HCF: ");
    scanf("%d", &num2);

    j = (num1<num2) ? num1 : num2;

    for(i=1; i<=j; i++)
{

    if(num1%i==0 && num2%i==0)
{
    hcf = i;
}
}

    printf("\nHCF of %d and %d is : %d\n\n", num1, num2,
hcf);
}
```

Input:

```
  HCF of two numbers:
 -----------------------
Input 1st number for HCF: 52413
Input 2nd number for HCF: 765732
```

```
HCF of 52413 and 765732 is : 3
```

44. Write a program in C to find LCM of any two numbers using HCF.

Source code:

```
#include <stdio.h>


void main()


{
    int i, num1, num2, j, hcf=1,lcm;



     printf("\n\n  LCM of two numbers:\n ");
     printf("----------------------\n");
    printf("Input 1st number for LCM: ");

    scanf("%d", &num1);

    printf("Input 2nd number for LCM: ");

    scanf("%d", &num2);


    j = (num1<num2) ? num1 : num2;


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

{


        if(num1%i==0 && num2%i==0)

{

        hcf = i;

}

}
```

```
    lcm=(num1*num2)/hcf;


    printf("\nThe LCM of %d and %d is : %d\n\n", num1,
num2, lcm);
}
```

Input:

```
LCM of two numbers:
 -----------------------
Input 1st number for LCM: 565
Input 2nd number for LCM: 343
```

Output:

```
The LCM of 565 and 343 is : 193795
```

45. Write a program in C to find LCM of any two numbers.

Source code:

```
#include <stdio.h>


void main()
{
    int i, num1, num2, max, lcm=1;



     printf("\n\n  LCM of two numbers:\n ");
     printf("----------------------\n");



    printf("Input 1st number for LCM: ");
    scanf("%d", &num1);
    printf("Input 2nd number for LCM: ");
    scanf("%d", &num2);
    max = (num1>num2) ? num1 : num2;



    for(i=max;  ; i+=max)
{

    if(i%num1==0 && i%num2==0)
{
    lcm = i;
    break;
}



}

    printf("\nLCM of %d and %d = %d\n\n", num1, num2,
lcm);
```

```
 }
```

Input:

```
  LCM of two numbers:
 ----------------------
Input 1st number for LCM: 45
Input 2nd number for LCM: 75
```

Output:

```
LCM of 45 and 75 = 225
```

46. Write a program in C to convert a binary number into a decimal number using math function.

Source code:

```
#include <stdio.h>
```

```
#include <math.h>

void main()


{

    int num1, n;

        int dec=0,i=0,j,d;


     printf("\n\nConvert Binary to Decimal:\n ");

     printf("------------------------\n");




        printf("Input  the binary number :");

        scanf("%d",&n);

        num1=n;

        while(n!=0)

{

           d = n % 10;
```

```
            dec=dec+d*pow(2,i);

            n=n/10;

            i++;

    }

          printf("\nThe Binary Number : %d\nThe equivalent
    Decimal  Number is : %d\n\n",num1,dec);

    }
```

Input:

```
Convert Binary to Decimal:

 -------------------------

Input  the binary number :625
```

Output:

```
The Binary Number : 625

The equivalent Decimal  Number is : 33
```

47. Write a C program to check whether a number is a Strong Number or not.

Source code:

```
#include <stdio.h>

void main()

{

    int i, n, num1, sum=0,j;

    long ft;


     printf("\n\n  Check whether a number is Strong
Number or not:\n ");

     printf("--------------------------------------------------\n");




    printf("Input a number to check whether it is Strong
number: ");

    scanf("%d", &n);
    num1 = n;


    for(j=n;j>0;j=j/10)
{


        ft = 1;
        for(i=1; i<=j % 10; i++)
{
            ft = ft * i;




}
         sum = sum + ft;


}
```

```
         if(sum==num1)

{

        printf("\n%d is Strong number.\n\n", num1);

}

         else

{

        printf("\n%d is not Strong number.\n\n", num1);

}




}
```

Input:

```
Check whether a number is Strong Number or not:

 -----------------------------------------------
```

```
Input a number to check whether it is Strong number:
145
```

Output:

```
145 is Strong number.
```

48. Write a C program to find Strong Numbers within a range of numbers.

Source code:

```
#include <stdio.h>
void main()
{
    int i, n, num1, sum=0,j,k,en,sn;
    long fact;

     printf("\n\n  Find Strong Numbers within an range of
numbers:\n ");
     printf("------------------------------------------------------\n");
```

```
   printf("Input starting range of number : ");

   scanf("%d", &sn);

   printf("Input ending range of number: ");

   scanf("%d", &en);

   printf("\n\nThe Strong numbers are :\n");


 for(k=sn;k<=en;k++)


{

    num1=k;

    sum=0;


   for(j=k;j>0;j=j/10)

{
        fact = 1;
         for(i=1; i<=j % 10; i++)
    {
          fact = fact * i;
    }
          sum = sum + fact;
}

    if(sum==num1)

        printf("%d  ", num1);
  }
        printf("\n\n");
}
```

Input:

```
  Find Strong Numbers within an range of numbers:
 -----------------------------------------------------
Input starting range of number : 1
Input ending range of number: 10000


Output:
The Strong numbers are :
1  2  145
```

49. Write a c program to find out the sum of an A.P. series.

Source code:

```
#include <stdio.h>
#include <math.h>

void main()
```

```
{

    int num1,diff,num2,i,ln;
    int sum=0;

     printf("\n\n  Find out the sum of A.P. series :\n ");
     printf("----------------------------------------\n");



    printf("Input  the starting number of the A.P. series: ");
    scanf("%d",&num1);

    printf("Input the number of items for  the A.P. series:
");
    scanf("%d",&num2);

    printf("Input  the common difference of A.P. series: ");
    scanf("%d",&diff);


    sum = ( num2 * ( 2 * num1 + ( num2 -1 ) * diff ) )/ 2;

    ln = num1 + (num2-1) * diff;

    printf("\nThe Sum of the  A.P. series are : \n");


    for(i=num1;i<=ln; i= i + diff ){

        if (i != ln)


            printf("%d + ",i);


        else


            printf("%d = %d \n\n",i,sum);

    }


}
```

Input:

```
  Find out the sum of A.P. series :
 ----------------------------------------
Input  the starting number of the A.P. series: 5
Input the number of items for  the A.P. series: 21
Input  the common difference of A.P. series: 3
```

Output:

```
The Sum of the  A.P. series are :
5 + 8 + 11 + 14 + 17 + 20 + 23 + 26 + 29 + 32 + 35 + 38 +
41 + 44 + 47 + 50 + 53 + 56 + 59 + 62 + 65 = 735
```

50. Write a program in C to convert a decimal number into octal without using an array.

Source code:

```
#include <stdio.h>

void main()

{

    int num, i, j, octno=0,dn;

    printf("\n\nConvert Decimal to Octal:\n ");
    printf("-------------------------\n");

    printf("Enter a number to convert : ");
    scanf("%d",&num);

    dn=num;
    i=1;
```

```
      for(j=num;j>0;j=j/8)
    {
        octno=octno+(j % 8)*i;
        i=i*10;
        num=num/8;
    }

     printf("\nThe Octal of %d is %d.\n\n",dn,octno);
}
```

Input:

```
Convert Decimal to Octal:
 -------------------------
Enter a number to convert : 653
```

Output:

```
The Octal of 653 is 1215.
```

51. Write a program in C to convert an octal number to a decimal without using an array.

Source code:

```
#include <stdio.h>


void main()

{

   int num, num2,p=1,k,ch=1;


       int dec=0,i=1,j,d;




     printf("\n\nConvert Octal to Decimal:\n ");
     printf("------------------------\n");



       printf("Input an octal number (using digit 0 - 7) :");
       scanf("%d",&num);


       num2=num;



    for(;num>0;num=num/10)


{
      k=num % 10;
      if(k>=8)
{
       ch=0;
  }
```

```
}

 switch(ch)

  {

  case 0 :
  printf("\nThe number is not an octal number. \n\n");
  break;

  case 1:

  num=num2;

      for (j=num;j>0;j=j/10)

      {
          d = j % 10;

            if(i==1)

             p=p*1;

            else

            p=p*8;


                dec=dec+(d*p);

                i++;

        }

        printf("\nThe Octal Number : %d\nThe equivalent
Decimal  Number : %d \n\n",num2,dec);

        break;

    }

}
```

Input:

Convert Octal to Decimal:

```
 --------------------------

Input an octal number (using digit 0 - 7) :2343
```

Output:

```
The Octal Number : 2343

The equivalent Decimal  Number : 1251
```

52. Write a program in c to find the Sum of GP series.

Source code:

```
#include <stdio.h>

#include <math.h>


void main()

{


    float gpf,cr,i,n,j;

    float ntrg,gpn;
    float sum=0;


     printf("\n\n Find the Sum of GP series.:\n ");
     printf("------------------------\n");


    printf("Input the first number of the G.P. series: ");
    scanf("%f",&gpf);


    printf("Input the number or terms in the G.P. series: ");
    scanf("%f",&ntrg);


    printf("Input the common ratio of G.P. series: ");
    scanf("%f",&cr);




     printf("\nThe numbers for the G.P. series:\n ");
         printf("%f ",gpf);
```

```
         sum=gpf;


     for(j=1;j<ntrg;j++)
   {
        gpn=gpf*pow(cr,j);
                sum=sum+gpn;
        printf("%f  ",gpn);
    }
    printf("\nThe Sum of the G.P. series : %f\n\n",sum);
}
```

Input:

```
 Find the Sum of GP series.:
 --------------------------
Input the first number of the G.P. series: 12
Input the number or terms in the G.P. series: 7
Input the common ratio of G.P. series: 4
```

Output:

```
The numbers for the G.P. series:
 12.000000 48.000000  192.000000  768.000000
3072.000000  12288.000000  49152.000000
The Sum of the G.P. series : 65532.000000
```

53. Write a program in C to convert a binary number to octal.

Source code:

```
#include <stdio.h>
#include <math.h>

void main()
{
```

```
int n1, num,p=1;
    int dec=0,i=1,j,d;
int bino=0,dn;



 printf("\n\nConvert Binary to Octal:\n ");
 printf("-------------------------\n");

    printf("Input a binary number :");
    scanf("%d",&num);
    n1=num;
    for (j=num;j>0;j=j/10)
    {
      d = j % 10;
        if(i==1)
        p=p*1;
        else
```

```
          p=p*2;


        dec=dec+(d*p);

        i++;

     }




  dn=dec;

  i=1;


   for(j=dec;j>0;j=j/8)

 {

     bino=bino+(j % 8)*i;

     i=i*10;

     num=num/8;

}
```

```
        printf("\nThe Binary Number : %d\nThe equivalent
Octal  Number : %d \n\n",n1,bino);
}
```

Input:

```
Convert Binary to Octal:
 -------------------------
Input a binary number :010111110
```

Output:

```
The Binary Number : 10111110
The equivalent Octal  Number : 276
```

54. Write a program in C to convert an octal number into binary.

Source code:

```
#include <stdio.h>

#include <math.h>


void main()

{




     long int n1, n2,p=1;

     long int dec=0,i=1,j,d;

     long int bino=0;




     printf("\n\nConvert Octal to Binary:\n ");

     printf("-------------------------\n");


        printf("Input an octal number (using digit 0 - 7) :");
        scanf("%ld",&n1);

        n2=n1;



   for (j=n1;j>0;j=j/10)

        {

          d = j % 10;

            if(i==1)

                   p=p*1;

            else

                  p=p*8;


           dec=dec+(d*p);

           i++;

        }
     i=1;


      for(j=dec;j>0;j=j/2)
       {
        bino=bino+(dec % 2)*i;
        i=i*10;
        dec=dec/2;
       }


        printf("\nThe Octal Number : %ld\nThe equivalent
Binary  Number : %ld \n\n",n2,bino);
}
```

Input:

```
Convert Octal to Binary:
 -------------------------
Input an octal number (using digit 0 - 7) :57
```

Output:

```
The Octal Number : 57

The equivalent Binary  Number : 101111
```

55. Write a program in C to convert a decimal number to hexadecimal.

Source code:

```
#include <stdio.h>


void main()

        {

        long int decn,rmd,q,dn=0,m,l;

        int i=1,j,tmp;

        char s;


       printf("\n\nConvert Decimal to Hexadecimal:\n ");
printf("Input  any Decimal number: ");
 scanf("%ld",&decn);
 q = decn;
 for(l=q;l>0;l=l/16)
        {
         tmp = l % 16;
         if( tmp < 10)
                    tmp =tmp + 48; else
                  tmp = tmp + 55;
                  dn=dn*100+tmp;
         }
   printf("\nThe equivalent Hexadecimal Number : ");
  for(m=dn;m>0;m=m/100)
     {
        s=m % 100;
        printf("%c",s);
     }
```

```
    printf("\n\n");
}
```

Output:

```
Convert Decimal to Hexadecimal:
 Input  any Decimal number: 23


The equivalent Hexadecimal Number : 17
```

56 Write a program in C to Check Whether a Number can be expressed as Sum of Two Prime Numbers.

Source code:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main()
{
```

```
int num,i,j,temp1,temp2,ctr=0;

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

scanf("%d",&num);

for(i=2;i<=num/2;i++){

temp1=i;

temp2=num-i;

for(j=2;j<=i/2;j++){

if(i%j==0){ctr++;break;}

}

if(ctr==0){

for(j=2;j<=(num-i)/2;j++){

if((num-i)%j==0){ctr++;break;}

}

if(ctr==0) printf("%d can be written as %d + %d.\n
",num,i,num-i);

}

ctr=0;
}
return 0;
}
```

Output:

```
input the number:16
16 can be written as 3 + 13.
 16 can be written as 5 + 11.
```

57Write a program in C to print a string in reverse order.

Source code:

```
#include <stdio.h>
#include <string.h>

void main()
{
   char str1[100], tmp;
   int l, lind, rind,i;
```

```
        printf("\n\nPrint a string in reverse order:\n ");



     printf("Input a string to reverse : ");
     scanf("%s", str1);
     l = strlen(str1);

     lind = 0;
     rind = l-1;

  for(i=lind;i<rind;i++)
        {
        tmp = str1[i];
        str1[i] = str1[rind];
        str1[rind] = tmp;
        rind--;
     }

     printf("Reversed string is: %s\n\n", str1);
```

```
}
```

Output:

```
Print a string in reverse order:
 Input a string to reverse : bvbhn
Reversed string is: nhbvb
```

58 Write a C program to find the length of a string without using the library function.

Source code:

```
#include <stdio.h>
#include <string.h>



void main()
{
   char str1[50];
   int i, l = 0;
```

```
      printf("\n\nFind the length of a string:\n ");



    printf("Input a string : ");
    scanf("%s", str1);

    for (i = 0; str1[i] != '\0'; i++)
    {
        l++;
    }
    printf("The string contains %d  number of characters. \n",l);
    printf("So, the length of the string %s is : %d\n\n", str1, l);
}
```

output:

```
Find the length of a string:
 Input a string : output
The string contains 6  number of characters.
```

So, the length of the string output is : 6

59 Write a program in C to check Armstrong number of n digits.

Source code:

```
#include <stdio.h>

#include <math.h>




int main()

{

   int n1, onum, r, result = 0, n = 0 ;

   printf("\n\n Check whether an n digits number is armstrong
or not :\n");




   printf(" Input  an integer : ");

   scanf("%d", &n1);
```

```
 onum = n1;

while (onum != 0)
{
    onum /= 10;
    ++n;
}

onum = n1;

while (onum != 0)
{
    r = onum % 10;
    result += pow(r, n);
    onum /= 10;
}
```

```
   if(result == n1)
      printf(" %d is an Armstrong number.\n\n", n1);
   else
      printf(" %d is not an Armstrong number.\n\n", n1);

   return 0;
}
```

Output:

```
 Check whether an n digits number is armstrong or not :
 Input  an integer : 123
 123 is not an Armstrong number.
```

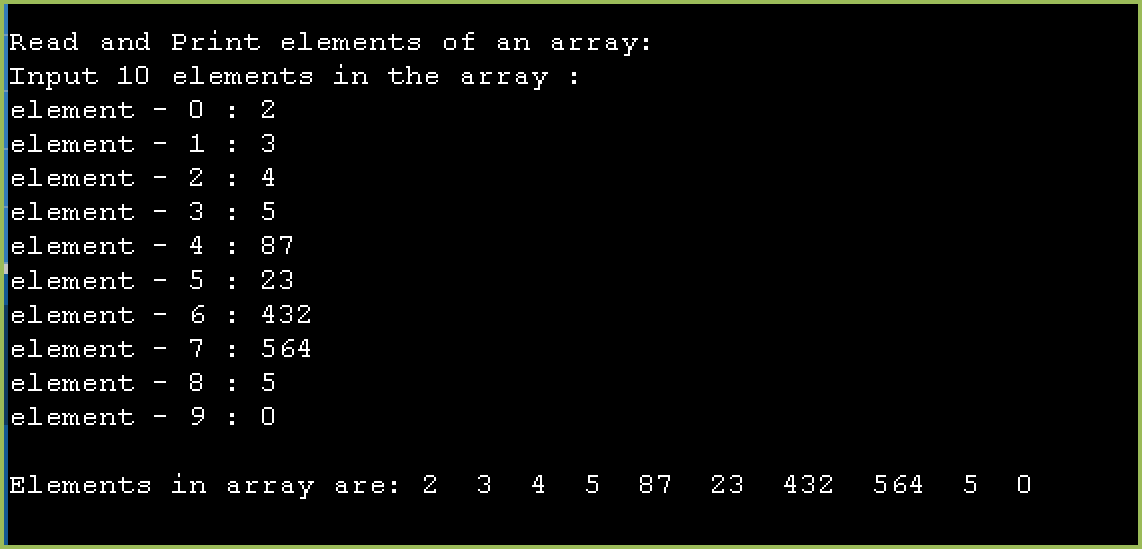
IMPLEMENTATION OF ARRAYS

1.Write a program in C to store elements in an array and print it.

Source code:

```
#include <stdio.h>


void  main()

{

   int arr[10];

   int i;

printf("\n\nRead and Print elements of an array:\n");

       printf("Input 10 elements in the array :\n");

for(i=0; i<10; i++)

   {

       printf("element - %d : ",i);

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

   }


printf("\nElements in array are: ");
for(i=0; i<10; i++)

   {

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

   }

   printf("\n");

}
```

Output:



```
Read and Print elements of an array:
Input 10 elements in the array :
element - 0 : 2
element - 1 : 3
element - 2 : 4
element - 3 : 5
element - 4 : 87
element - 5 : 23
element - 6 : 432
element - 7 : 564
element - 8 : 5
element - 9 : 0

Elements in array are: 2   3   4   5   87   23   432   564   5   0
```

2.Write a program in C to read n number of values in an array and display it in reverse order.

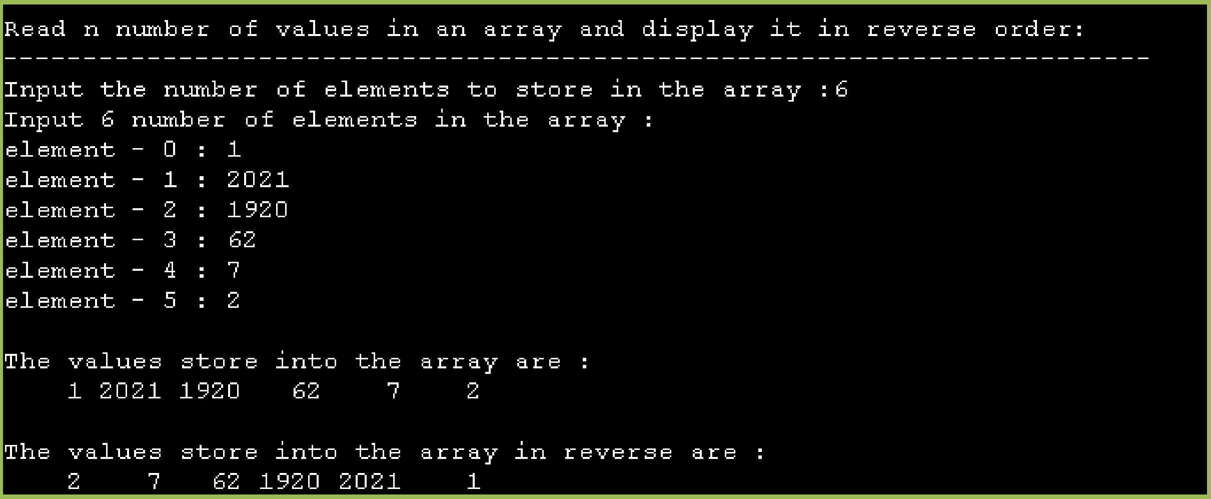
Ans

Source code:

```
#include <stdio.h>

void main()
{
   int i,n,a[100];

printf("\n\nRead n number of values in an array and display it
in reverse order:\n");
       printf("---------------------------------------------------------------------
--\n");

printf("Input the number of elements to store in the array :");
   scanf("%d",&n);

printf("Input %d number of elements in the array :\n",n);
   for(i=0;i<n;i++)
      {
         printf("element - %d : ",i);
```

```
          scanf("%d",&a[i]);
          }
        printf("\nThe values store into the array are : \n");
  for(i=0;i<n;i++)
   {
        printf("% 5d",a[i]);
         }

printf("\n\nThe values store into the array in reverse are :\n");
  for(i=n-1;i>=0;i--)
    {
        printf("% 5d",a[i]);
          }
  printf("\n\n");
}
```

Output:



```
Read n number of values in an array and display it in reverse order:
------------------------------------------------------------------------
Input the number of elements to store in the array :6
Input 6 number of elements in the array :
element - 0 : 1
element - 1 : 2021
element - 2 : 1920
element - 3 : 62
element - 4 : 7
element - 5 : 2

The values store into the array are :
    1 2021 1920    62     7     2

The values store into the array in reverse are :
    2     7    62 1920 2021     1
```

3.Write a program in C to find the sum of all elements of the array.

Ans

Source code:

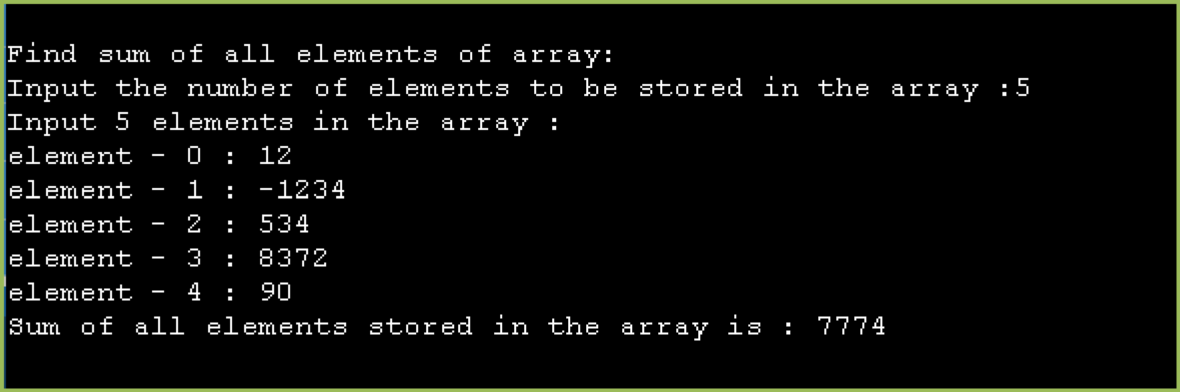
```
#include <stdio.h>


void main()

{

   int a[100];

   int i, n, sum=0;
```

```
printf("\n\nFind sum of all elements of array:\n");



printf("Input the number of elements to be stored in the array
:");
       scanf("%d",&n);


printf("Input %d elements in the array :\n",n);
       for(i=0;i<n;i++)
        {
         printf("element - %d : ",i);
               scanf("%d",&a[i]);
             }


for(i=0; i<n; i++)
    {
         sum += a[i];
    }
```

printf("Sum of all elements stored in the array is : %d\n\n", sum);

}

Output:



```
Find sum of all elements of array:
Input the number of elements to be stored in the array :5
Input 5 elements in the array :
element - 0 : 12
element - 1 : -1234
element - 2 : 534
element - 3 : 8372
element - 4 : 90
Sum of all elements stored in the array is : 7774
```

4. Write a program in C to copy the elements of one array into another array.

Ans

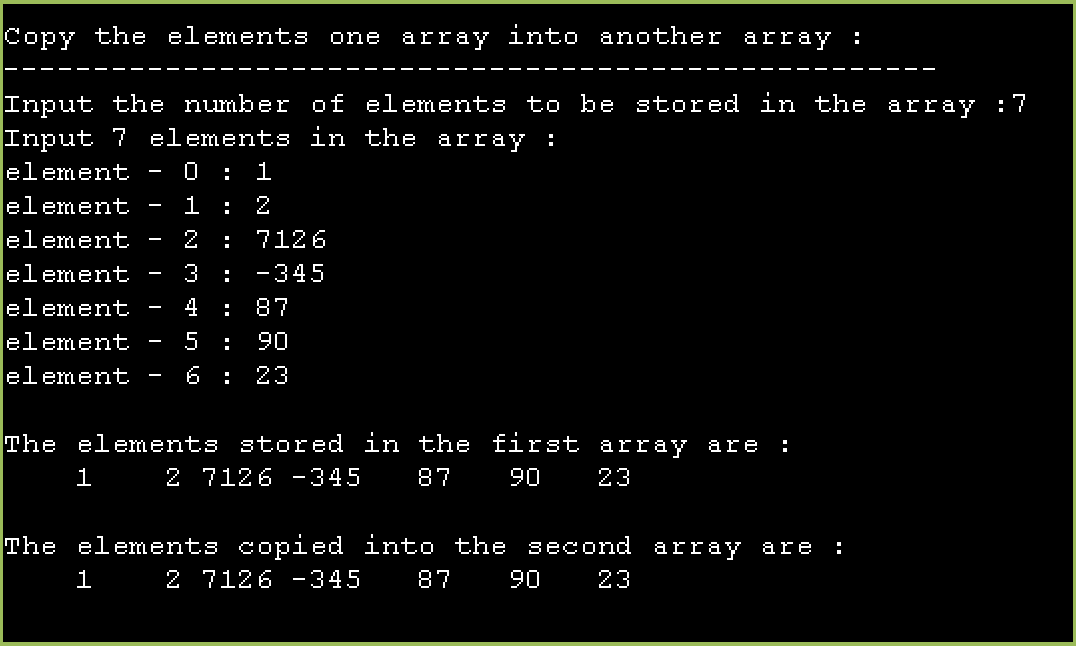
Source code:

```
#include <stdio.h>


void main()

{

   int arr1[100], arr2[100];
```

```
    int i, n;



printf("\n\nCopy the elements one array into another array
:\n");
       printf("--------------------------------------------------\n");

printf("Input the number of elements to be stored in the array
:");
       scanf("%d",&n);

printf("Input %d elements in the array :\n",n);
       for(i=0;i<n;i++)
        {
         printf("element - %d : ",i);
               scanf("%d",&arr1[i]);
             }
    /* Copy elements of first array into second array.*/
for(i=0; i<n; i++)
```

```
    {
        arr2[i] = arr1[i];
    }

    /* Prints the elements of first array   */
printf("\nThe elements stored in the first array are :\n");
for(i=0; i<n; i++)
    {
printf("% 5d", arr1[i]);
    }
printf("\n\nThe elements copied into the second array are
:\n");
for(i=0; i<n; i++)
    {
printf("% 5d", arr2[i]);
    }
                printf("\n\n");
}
```

Output:



```
Copy the elements one array into another array :
----------------------------------------------------
Input the number of elements to be stored in the array :7
Input 7 elements in the array :
element - 0 : 1
element - 1 : 2
element - 2 : 7126
element - 3 : -345
element - 4 : 87
element - 5 : 90
element - 6 : 23

The elements stored in the first array are :
    1    2 7126 -345   87   90   23

The elements copied into the second array are :
    1    2 7126 -345   87   90   23
```

5. Write a program in C to count a total number of duplicate elements in an array.

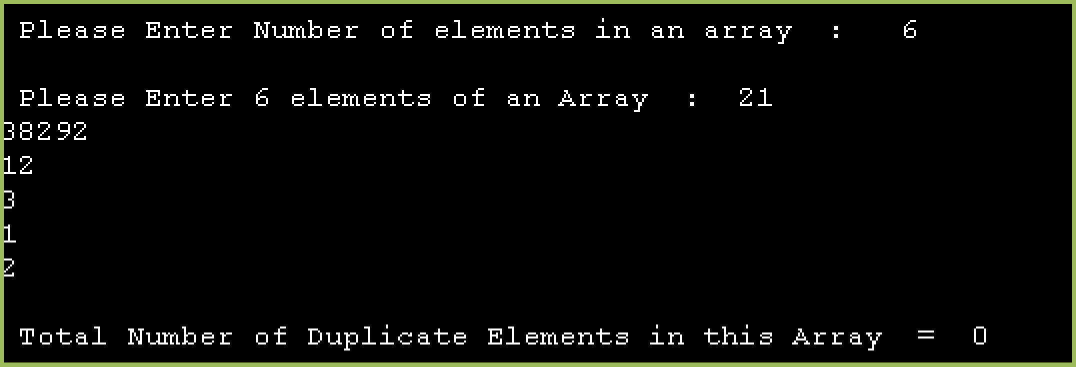
Ans

Source code:

```
#include <stdio.h>


int main()

{

     int arr[10], i, j, Size, Count = 0;
```

```
        printf("\n Please Enter Number of elements in an array  :
");
        scanf("%d", &Size);


        printf("\n Please Enter %d elements of an Array  :  ", Size);
        for (i = 0; i < Size; i++)
        {
        scanf("%d", &arr[i]);
        }


        for (i = 0; i < Size; i++)
        {
                for(j = i + 1; j < Size; j++)
                {
                if(arr[i] == arr[j])
                {
                        Count++;
                                break;
                        }
        }
    }

    printf("\n Total Number of Duplicate Elements in this Array
=  %d ", Count);

    return 0;
}
```

Output:



```
 Please Enter Number of elements in an array  :   6

 Please Enter 6 elements of an Array  :  21
38292
12
3
1
2

 Total Number of Duplicate Elements in this Array  =  0
```

6. Write a program in C to print all unique elements in an array.

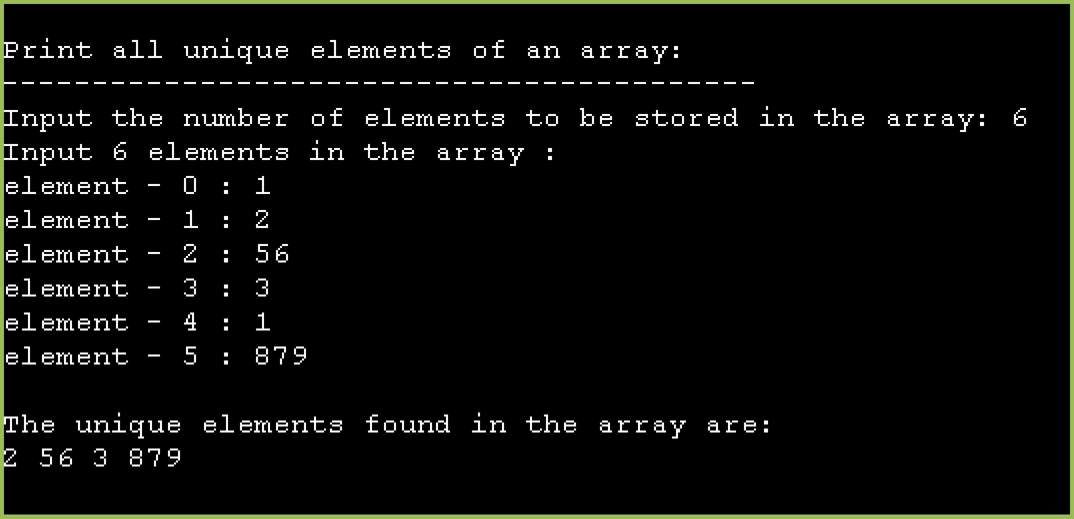
Ans

Source code:

```
#include <stdio.h>
```

```
int main()

{

    int arr1[100], n,ctr=0;

    int i, j, k;

printf("\n\nPrint all unique elements of an array:\n");

       printf("------------------------------------------\n");

printf("Input the number of elements to be stored in the array:
");

       scanf("%d",&n);

printf("Input %d elements in the array :\n",n);

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

            {

         printf("element - %d : ",i);

               scanf("%d",&arr1[i]);

             }

printf("\nThe unique elements found in the array are: \n");

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

    {

        ctr=0;
for(j=0,k=n; j<k+1; j++)
        {
            /*Increment the counter when the seaarch value is
duplicate.*/
            if (i!=j)
            {
                        if(arr1[i]==arr1[j])
              {
                 ctr++;
               }
             }
        }
       if(ctr==0)
        {
printf("%d ",arr1[i]);
        }
    }
       printf("\n\n");
}
```

Output:



```
Print all unique elements of an array:
------------------------------------------
Input the number of elements to be stored in the array: 6
Input 6 elements in the array :
element - 0 : 1
element - 1 : 2
element - 2 : 56
element - 3 : 3
element - 4 : 1
element - 5 : 879

The unique elements found in the array are:
2 56 3 879
```

7. Write a program in C to merge two arrays of same size sorted in decending order.

Ans

Source code:

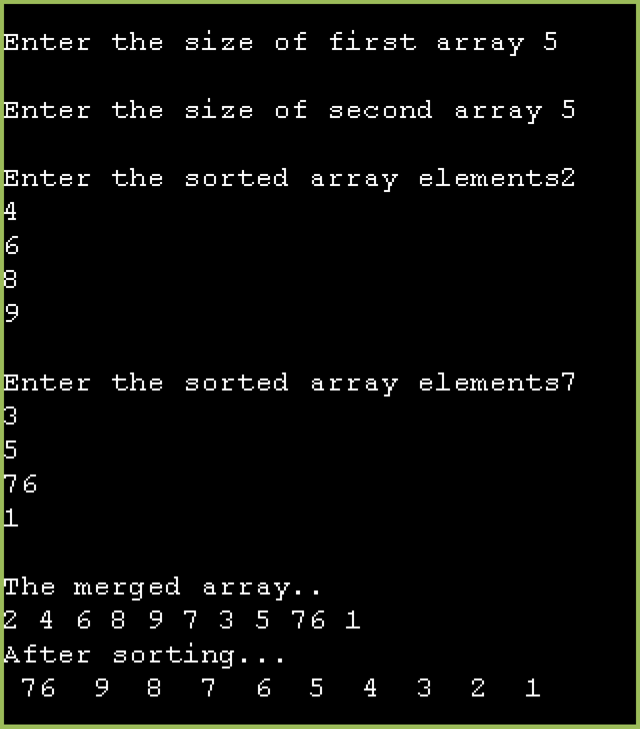
```
#include <stdio.h>


int main()

{

   int n1,n2,n3;

printf("\nEnter the size of first array ");

   scanf("%d",&n1);
```

```
printf("\nEnter the size of second array ");
    scanf("%d",&n2);


    n3=n1+n2;
printf("\nEnter the sorted array elements");
    int a[n1],b[n2],c[n3];
for(int i=0;i<n1;i++)         {
       scanf("%d",&a[i]);
       c[i]=a[i];
    }
    int k=n1;
printf("\nEnter the sorted array elements");
for(int i=0;i<n2;i++)
    {
        scanf("%d",&b[i]);
        c[k]=b[i];
        k++;
    }
```

```
printf("\nThe merged array..\n");
for(int i=0;i<n3;i++)
printf("%d ",c[i]);


printf("\nAfter sorting...\n");
for(int i=0;i<n3;i++)
  {
    int temp;
for(int j=i+1; j<n3 ;j++)
    {
      if(c[i]<c[j])
      {
        temp=c[i];
        c[i]=c[j];
        c[j]=temp;
      }
    }
  }
```

```
for(int i=0 ; i<n3 ; i++)
    {
printf(" %d ",c[i]);
    }
    return 0;
}
```

Output:



```
Enter the size of first array 5

Enter the size of second array 5

Enter the sorted array elements2
4
6
8
9

Enter the sorted array elements7
3
5
76
1

The merged array..
2 4 6 8 9 7 3 5 76 1
After sorting...
 76  9  8  7  6  5  4  3  2  1
```

8.Write a program in C to count the frequency of each element of an array.

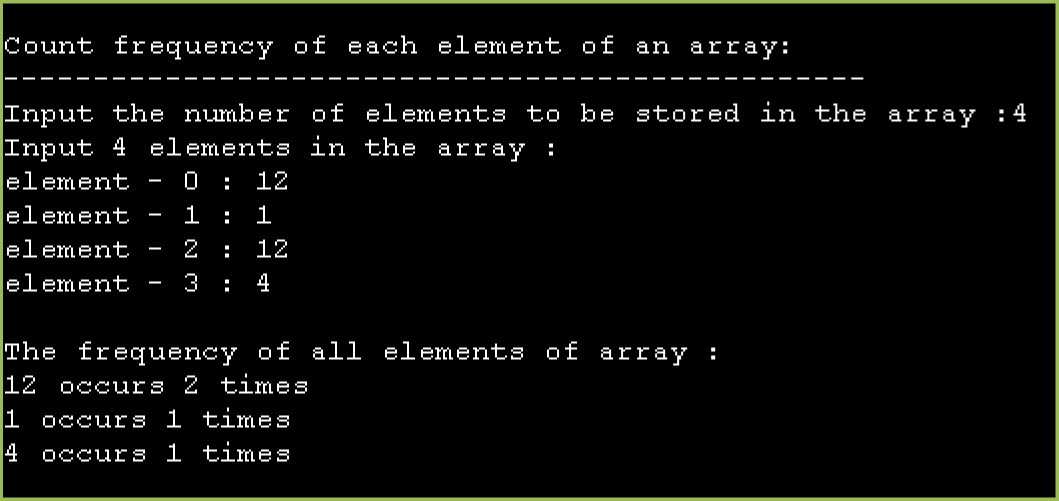
Ans

Source code:

```
#include <stdio.h>

void main()
{
   int arr1[100], fr1[100];
   int n, i, j, ctr;



printf("\n\nCount frequency of each element of an array:\n");
      printf("-------------------------------------------------\n");

printf("Input the number of elements to be stored in the array :");
      scanf("%d",&n);

printf("Input %d elements in the array :\n",n);
      for(i=0;i<n;i++)
```

```
            {
         printf("element - %d : ",i);
            scanf("%d",&arr1[i]);
               fr1[i] = -1;
            }
for(i=0; i<n; i++)
    {
        ctr = 1;
for(j=i+1; j<n; j++)
        {
            if(arr1[i]==arr1[j])
            {
                ctr++;
                fr1[j] = 0;
            }
        }

        if(fr1[i]!=0)
        {
           fr1[i] = ctr;
        }
    }
printf("\nThe frequency of all elements of array : \n");
for(i=0; i<n; i++)
    {
        if(fr1[i]!=0)
        {
printf("%d occurs %d times\n", arr1[i], fr1[i]);
        }
    }
}
```

Output:


```
Count frequency of each element of an array:
------------------------------------------------
Input the number of elements to be stored in the array :4
Input 4 elements in the array :
element - 0 : 12
element - 1 : 1
element - 2 : 12
element - 3 : 4

The frequency of all elements of array :
12 occurs 2 times
1 occurs 1 times
4 occurs 1 times
```

9. Write a program in C to find the maximum and minimum element in an array.

Ans

Source code:

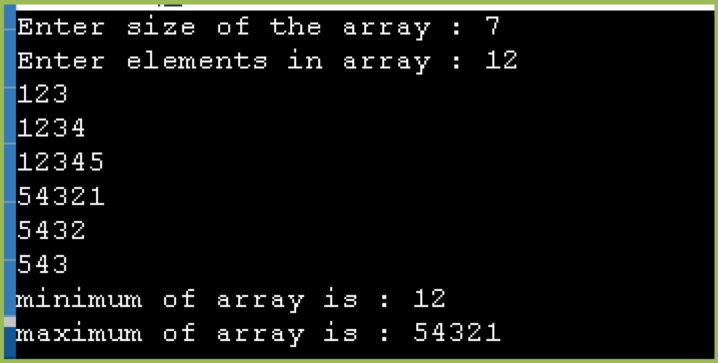
```
#include<stdio.h>

#include <conio.h>


int main()

{

   int a[1000],i,n,min,max;


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

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


printf("Enter elements in array : ");
for(i=0; i<n; i++)
    {
        scanf("%d",&a[i]);
    }


    min=max=a[0];
for(i=1; i<n; i++)
    {
         if(min>a[i])
                    min=a[i];
                     if(max<a[i])
                      max=a[i];
    }
printf("minimum of array is : %d",min);
printf("\nmaximum of array is : %d",max);
```

    return 0;

}

Output:



```
Enter size of the array : 7
Enter elements in array : 12
123
1234
12345
54321
5432
543
minimum of array is : 12
maximum of array is : 54321
```

10. Write a program in C to separate odd and even integers in separate arrays.

Ans

Source code:

#include <stdio.h>

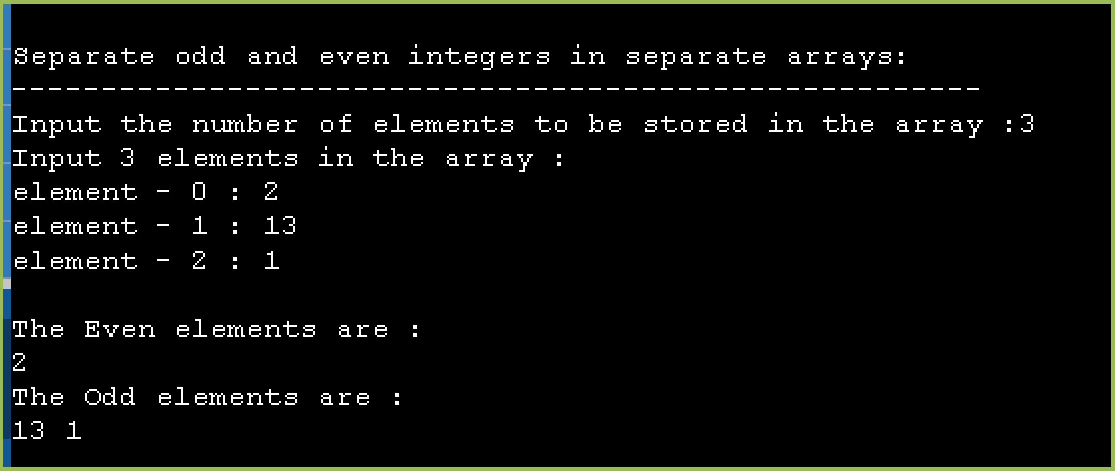
void main()

 {

```
    int arr1[10], arr2[10], arr3[10];

    int i,j=0,k=0,n;



printf("\n\nSeparate odd and even integers in separate arrays:\n");

       printf("--------------------------------------------------\n");


printf("Input the number of elements to be stored in the array :");

       scanf("%d",&n);


printf("Input %d elements in the array :\n",n);

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

            {

        printf("element - %d : ",i);

              scanf("%d",&arr1[i]);

             }
    for(i=0;i<n;i++)
    {
        if (arr1[i]%2 == 0)
        {
            arr2[j] = arr1[i];
           j++;
        }
        else
        {
           arr3[k] = arr1[i];
           k++;
        }
    }

printf("\nThe Even elements are : \n");
    for(i=0;i<j;i++)
    {
        printf("%d ",arr2[i]);
```

```
   }


printf("\nThe Odd elements are :\n");
   for(i=0;i<k;i++)
   {
       printf("%d ", arr3[i]);
   }
   printf("\n\n");
 }
```

Output:



```
Separate odd and even integers in separate arrays:
------------------------------------------------------
Input the number of elements to be stored in the array :3
Input 3 elements in the array :
element - 0 : 2
element - 1 : 13
element - 2 : 1

The Even elements are :
2
The Odd elements are :
13 1
```

11. Write a program in C to sort elements of array in ascending order.

Ans

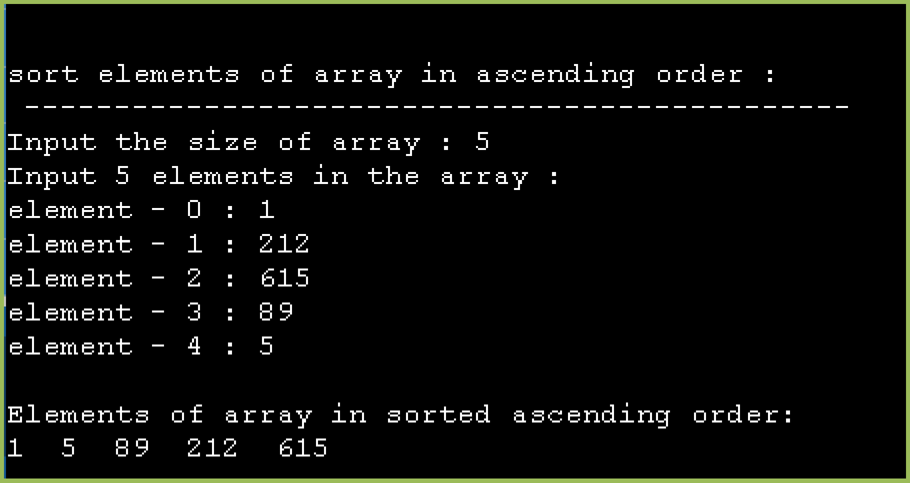
Source code:

```
#include <stdio.h>


void main()

{

   int arr1[100];

   int n, i, j, tmp;




printf("\n\nsort elements of array in ascending order :\n ");

      printf("-------------------------------------------------\n");


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

scanf("%d", &n);


printf("Input %d elements in the array :\n",n);

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

           {
```

```
         printf("element - %d : ",i);
              scanf("%d",&arr1[i]);
            }

for(i=0; i<n; i++)
    {
for(j=i+1; j<n; j++)
        {
            if(arr1[j] <arr1[i])
            {
                tmp = arr1[i];
                arr1[i] = arr1[j];
                arr1[j] = tmp;
            }
        }
    }
printf("\nElements of array in sorted ascending order:\n");
for(i=0; i<n; i++)
```

```
   {
printf("%d  ", arr1[i]);
   }
              printf("\n\n");
}
```

Output:



```
sort elements of array in ascending order :
 ----------------------------------------------
Input the size of array : 5
Input 5 elements in the array :
element - 0 : 1
element - 1 : 212
element - 2 : 615
element - 3 : 89
element - 4 : 5

Elements of array in sorted ascending order:
1  5  89  212  615
```

12. Write a program in C to sort elements of the array in descending order.

Ans

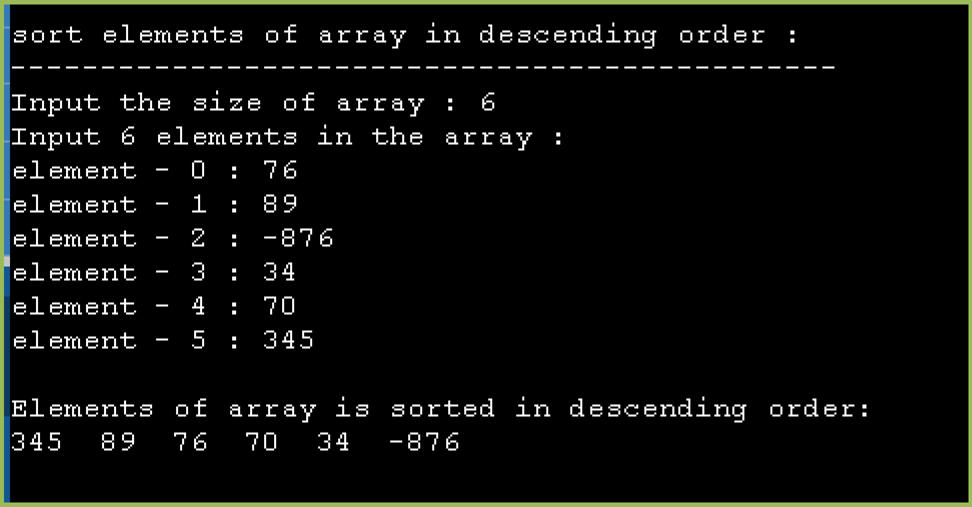
Source code:

```
#include <stdio.h>


void main()
```

```
{
   int arr1[100];
   int n, i, j, tmp;
printf("\n\nsort elements of array in descending order :\n");
     printf("-------------------------------------------\n");

printf("Input the size of array : ");
scanf("%d", &n);

printf("Input %d elements in the array :\n",n);
     for(i=0;i<n;i++)
         {
      printf("element - %d : ",i);
          scanf("%d",&arr1[i]);
         }
for(i=0; i<n; i++)
   {
for(j=i+1; j<n; j++)
```

```
        {
            if(arr1[i] < arr1[j])
            {
                tmp = arr1[i];
                arr1[i] = arr1[j];
                arr1[j] = tmp;
            }
        }
    }

printf("\nElements of array is sorted in descending order:\n");
for(i=0; i<n; i++)
    {
printf("%d  ", arr1[i]);
    }
                 printf("\n\n");
}
```

Output:


```
sort elements of array in descending order :
---------------------------------------------
Input the size of array : 6
Input 6 elements in the array :
element - 0 : 76
element - 1 : 89
element - 2 : -876
element - 3 : 34
element - 4 : 70
element - 5 : 345

Elements of array is sorted in descending order:
345  89  76  70  34  -876
```

13. Write a program in C to insert New value in the array (sorted list)

Ans

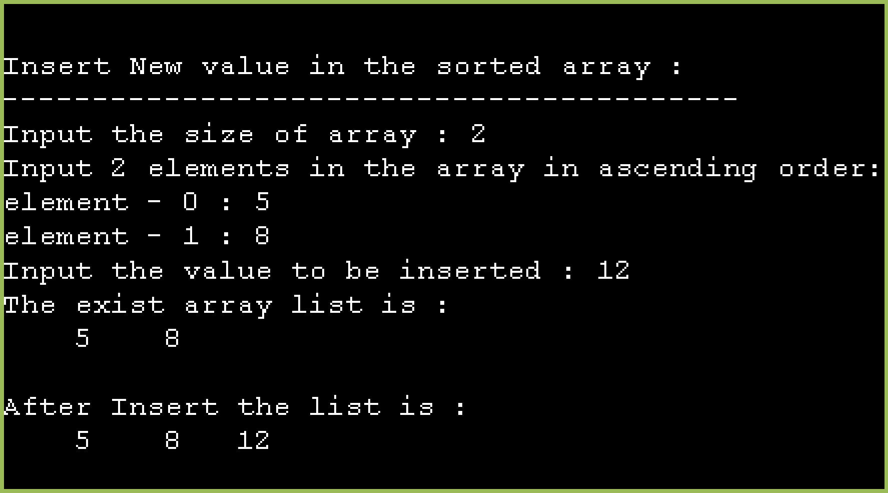
Source code:

```
#include <stdio.h>


int main()

{

  int arr1[100],i,n,p,inval;

printf("\n\nInsert New value in the sorted array :\n");

     printf("-----------------------------------------\n");

printf("Input the size of array : ");
```

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

/* Stored values into the array*/

printf("Input %d elements in the array in ascending
order:\n",n);

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

            {

         printf("element - %d : ",i);

               scanf("%d",&arr1[i]);

             }

printf("Input the value to be inserted : ");

   scanf("%d",&inval);

printf("The exist array list is :\n");

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

printf("% 5d",arr1[i]);

   /* Determine the position where the new value will be
insert.*/

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

   {
    if(inval<arr1[i])
     {
       p = i;
       break;
     }
     else
     {
        p=i+1;
      }
          }
   /* move all data at right side of the array */
for(i=n+1;i>=p;i--)
      arr1[i]= arr1[i-1];
   /* insert value at the proper position */
      arr1[p]=inval;
printf("\n\nAfter Insert the list is :\n");
   for(i=0;i<=n;i++)
printf("% 5d",arr1[i]);
```

```
        printf("\n");
}
```

Output:



```
Insert New value in the sorted array :
-----------------------------------------
Input the size of array : 2
Input 2 elements in the array in ascending order:
element - 0 : 5
element - 1 : 8
Input the value to be inserted : 12
The exist array list is :
    5    8

After Insert the list is :
    5    8   12
```

14. Write a program in C to insert New value in the array (unsorted list).

Ans

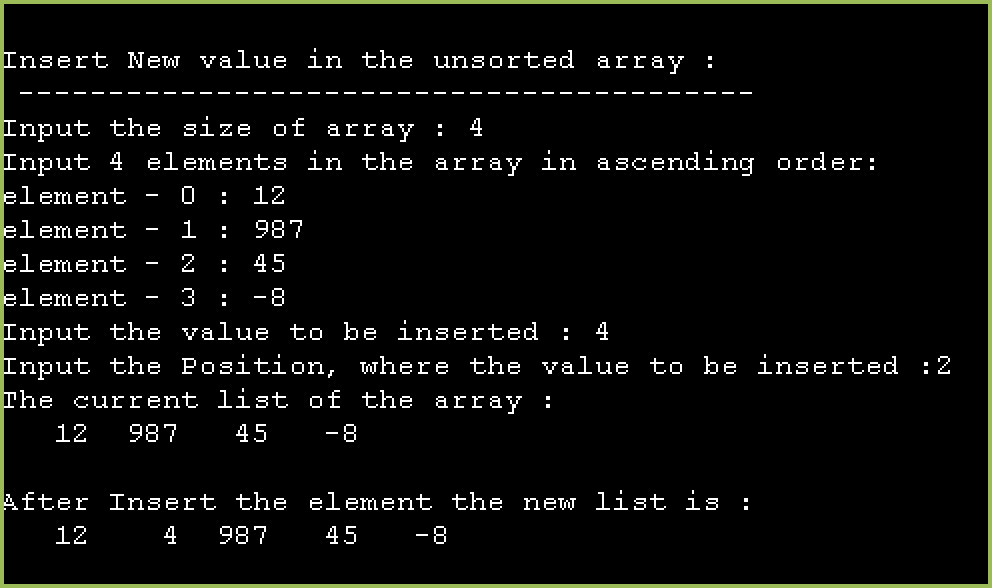
Source code:

```
#include <stdio.h>


void main()
{
  int arr1[100],i,n,p,x;
```

```
printf("\n\nInsert New value in the unsorted array : \n ");
      printf("---------------------------------------\n");
printf("Input the size of array : ");
scanf("%d", &n);
   /* Stored values into the array*/
printf("Input %d elements in the array in ascending
order:\n",n);
      for(i=0;i<n;i++)
          {
        printf("element - %d : ",i);
             scanf("%d",&arr1[i]);
           }



printf("Input the value to be inserted : ");
  scanf("%d",&x);
printf("Input the Position, where the value to be inserted :");
  scanf("%d",&p);
```

```
printf("The current list of the array :\n");
  for(i=0;i<n;i++)
printf("% 5d",arr1[i]);
  /* Move all data at right side of the array */
  for(i=n;i>=p;i--)
    arr1[i]= arr1[i-1];
  /* insert value at given position */
    arr1[p-1]=x;



printf("\n\nAfter Insert the element the new list is :\n");
  for(i=0;i<=n;i++)
printf("% 5d",arr1[i]);
        printf("\n\n");
}
```

Output:


```
Insert New value in the unsorted array :
 -----------------------------------------
Input the size of array : 4
Input 4 elements in the array in ascending order:
element - 0 : 12
element - 1 : 987
element - 2 : 45
element - 3 : -8
Input the value to be inserted : 4
Input the Position, where the value to be inserted :2
The current list of the array :
   12   987    45    -8

After Insert the element the new list is :
   12     4   987    45    -8
```

15. Write a program in C to delete an element at desired position from an array.

Ans

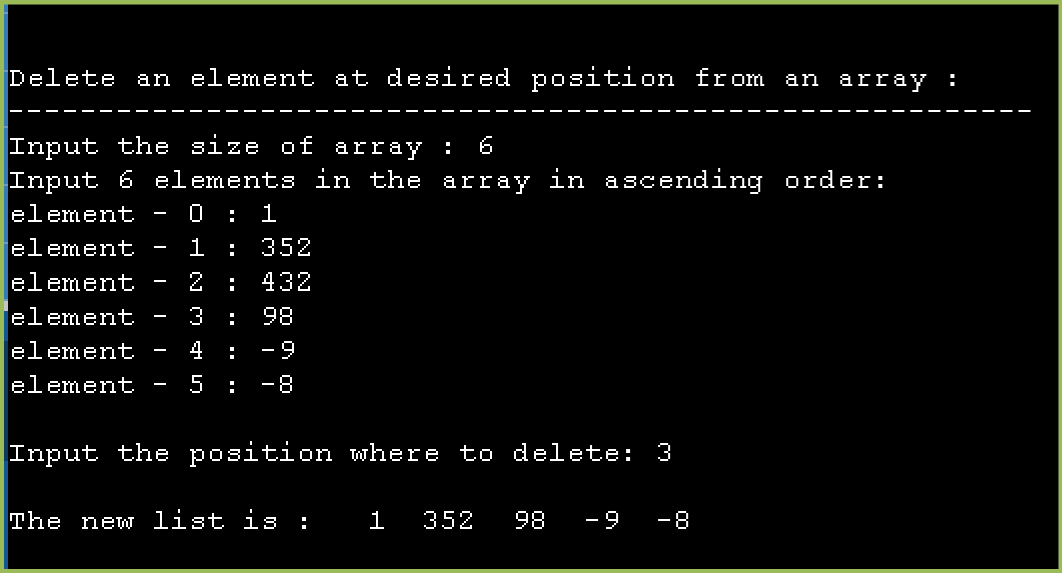
Source code:

```
#include <stdio.h>


void main(){

  int arr1[50],i,pos,n;


printf("\n\nDelete an element at desired position from an array :\n");
```

```
       printf("------------------------------------------------------\n");


    printf("Input the size of array : ");
    scanf("%d", &n);
       /* Stored values into the array*/
    printf("Input %d elements in the array in ascending
    order:\n",n);
       for(i=0;i<n;i++)
           {
        printf("element - %d : ",i);
             scanf("%d",&arr1[i]);
            }


    printf("\nInput the position where to delete: ");
     scanf("%d",&pos);
    /*---- locate the position of i in the array -------*/
     i=0;
     while(i!=pos-1)
            i++;
```

```
/*---- the position of i in the array will be replaced by the
      value of its right */
  while(i<n){
          arr1[i]=arr1[i+1];
          i++;
  }
  n--;
printf("\nThe new list is : ");
  for(i=0;i<n;i++)
       {
printf("  %d",arr1[i]);
                }

}
```

Output:



```
Delete an element at desired position from an array :
---------------------------------------------------------
Input the size of array : 6
Input 6 elements in the array in ascending order:
element - 0 : 1
element - 1 : 352
element - 2 : 432
element - 3 : 98
element - 4 : -9
element - 5 : -8

Input the position where to delete: 3

The new list is :    1   352   98   -9   -8
```

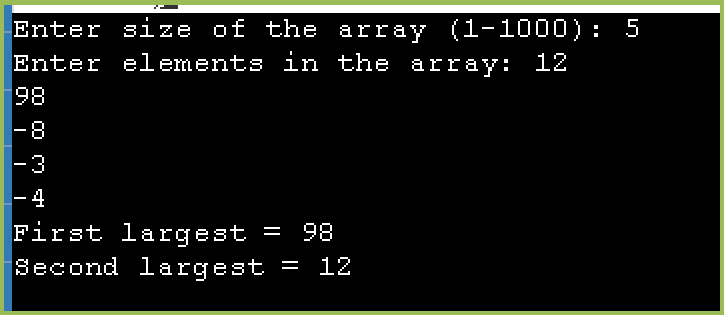
16. Write a program in C to find the second largest element in an array.

Ans

Source code:

```
#include <stdio.h>

#include <limits.h>


#define MAX_SIZE 1000


int main()

{
    int arr[MAX_SIZE], size, i;

    int max1, max2;

printf("Enter size of the array (1-1000): ");

scanf("%d", &size);

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

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

    {

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

    }

    max1 = max2 = INT_MIN;

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

    {

        if(arr[i] > max1)

        {

            max2 = max1;

            max1 = arr[i];

        }

        else if(arr[i] > max2 && arr[i] < max1)
        {
            max2 = arr[i];
        }
    }

printf("First largest = %d\n", max1);
printf("Second largest = %d", max2);

    return 0;
}
```

Output:



```
Enter size of the array (1-1000): 5
Enter elements in the array: 12
98
-8
-3
-4
First largest = 98
Second largest = 12
```

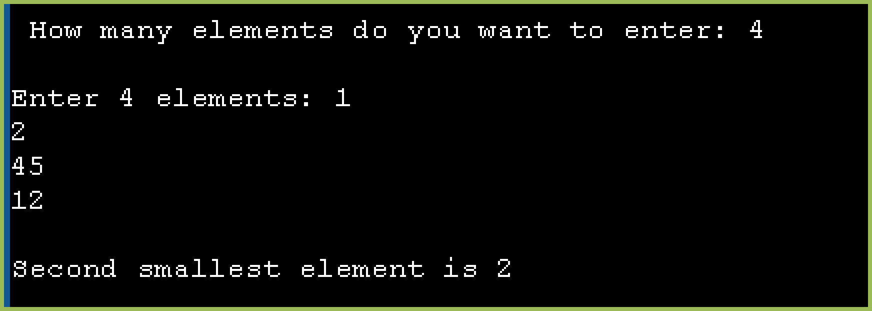
17. Write a program in C to find the second smallest element in an array.

Ans

Source code:

```
#include <stdio.h>

#include <string.h>


int main()

{

   int smallest, secondsmallest;

   int array[100], size, i;

printf("\n How many elements do you want to enter: ");

scanf("%d", &size);

printf("\nEnter %d elements: ", size);

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

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

   if (array[0] < array[1]) {

      smallest = array[0];

      secondsmallest = array[1];

   }

   else {
      smallest = array[1];

      secondsmallest = array[0];

    }

    for (i = 2; i < size; i++) {

        if (array[i] < smallest) {

        secondsmallest = smallest;

        smallest = array[i];

        }

        else if (array[i] < secondsmallest) {

            secondsmallest = array[i];

        }

    }

printf(" \nSecond smallest element is %d", secondsmallest);

}
```

Output:



```
 How many elements do you want to enter: 4

Enter 4 elements: 1
2
45
12

Second smallest element is 2
```

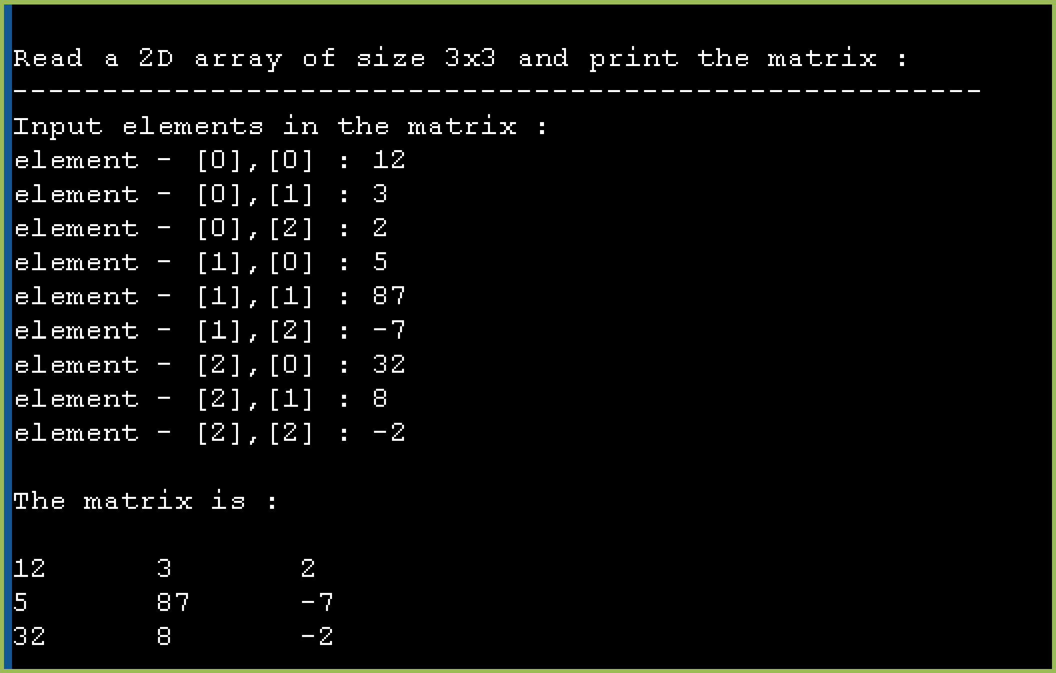
18. Write a program in C for a 2D array of size 3x3 and print the matrix.

Ans

Source code:

```
#include <stdio.h>


void main()

{

  int arr1[3][3],i,j;


printf("\n\nRead a 2D array of size 3x3 and print the matrix
:\n");

       printf("----------------------------------------------------\n");




printf("Input elements in the matrix :\n");

  for(i=0;i<3;i++)

  {

      for(j=0;j<3;j++)
      {
        printf("element - [%d],[%d] : ",i,j);
              scanf("%d",&arr1[i][j]);
      }
  }

printf("\nThe matrix is : \n");
for(i=0;i<3;i++)
  {
      printf("\n");
      for(j=0;j<3;j++)
           printf("%d\t",arr1[i][j]);
  }
 printf("\n\n");
}
```

Output:


```
Read a 2D array of size 3x3 and print the matrix :
------------------------------------------------------
Input elements in the matrix :
element - [0],[0] : 12
element - [0],[1] : 3
element - [0],[2] : 2
element - [1],[0] : 5
element - [1],[1] : 87
element - [1],[2] : -7
element - [2],[0] : 32
element - [2],[1] : 8
element - [2],[2] : -2

The matrix is :

12      3       2
5       87      -7
32      8       -2
```

19. Write a program in C for addition of two Matrices of same size.

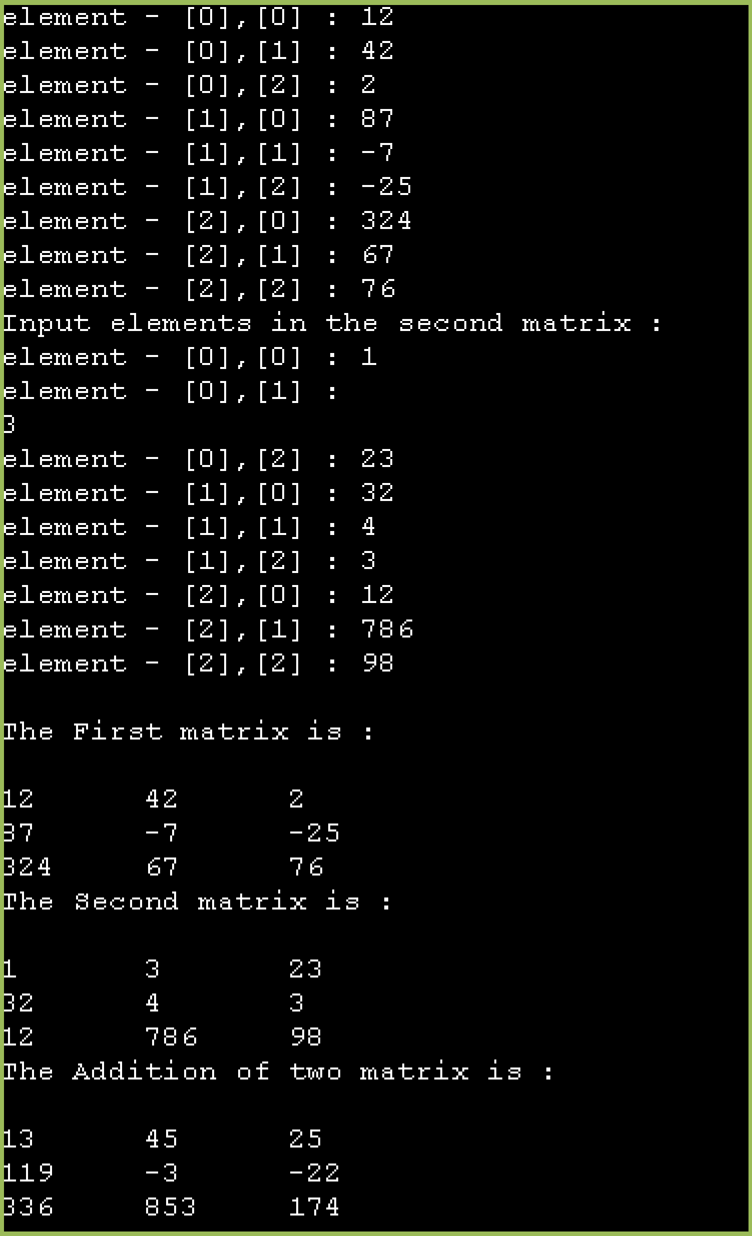
Ans

Source code:

```
#include <stdio.h>


void main()

{

  int arr1[50][50],brr1[50][50],crr1[50][50],i,j,n;
```

```
printf("\n\nAddition of two Matrices :\n");
       printf("------------------------------\n");
printf("Input the size of the square matrix (less than 5): ");
scanf("%d", &n);
printf("Input elements in the first matrix :\n");
       for(i=0;i<n;i++)
        {
            for(j=0;j<n;j++)
            {
         printf("element - [%d],[%d] : ",i,j);
                    scanf("%d",&arr1[i][j]);
            }
        }

printf("Input elements in the second matrix :\n");
       for(i=0;i<n;i++)
        {
            for(j=0;j<n;j++)
            {
         printf("element - [%d],[%d] : ",i,j);
                  scanf("%d",&brr1[i][j]);
            }
        }
printf("\nThe First matrix is :\n");
  for(i=0;i<n;i++)
    {
      printf("\n");
      for(j=0;j<n;j++)
           printf("%d\t",arr1[i][j]);
    }

printf("\nThe Second matrix is :\n");
  for(i=0;i<n;i++)
    {
      printf("\n");
      for(j=0;j<n;j++)
      printf("%d\t",brr1[i][j]);
    }
   for(i=0;i<n;i++)
       for(j=0;j<n;j++)
            crr1[i][j]=arr1[i][j]+brr1[i][j];
printf("\nThe Addition of two matrix is : \n");
   for(i=0;i<n;i++){
       printf("\n");
       for(j=0;j<n;j++)
            printf("%d\t",crr1[i][j]);
   }
   printf("\n\n");
}
```

Output:



```
element - [0],[0] : 12
element - [0],[1] : 42
element - [0],[2] : 2
element - [1],[0] : 87
element - [1],[1] : -7
element - [1],[2] : -25
element - [2],[0] : 324
element - [2],[1] : 67
element - [2],[2] : 76
Input elements in the second matrix :
element - [0],[0] : 1
element - [0],[1] :
3
element - [0],[2] : 23
element - [1],[0] : 32
element - [1],[1] : 4
element - [1],[2] : 3
element - [2],[0] : 12
element - [2],[1] : 786
element - [2],[2] : 98

The First matrix is :

12      42      2
87      -7      -25
324     67      76
The Second matrix is :

1       3       23
32      4       3
12      786     98
The Addition of two matrix is :

13      45      25
119     -3      -22
336     853     174
```

20. Write a program in C for subtraction of two Matrices.

Ans

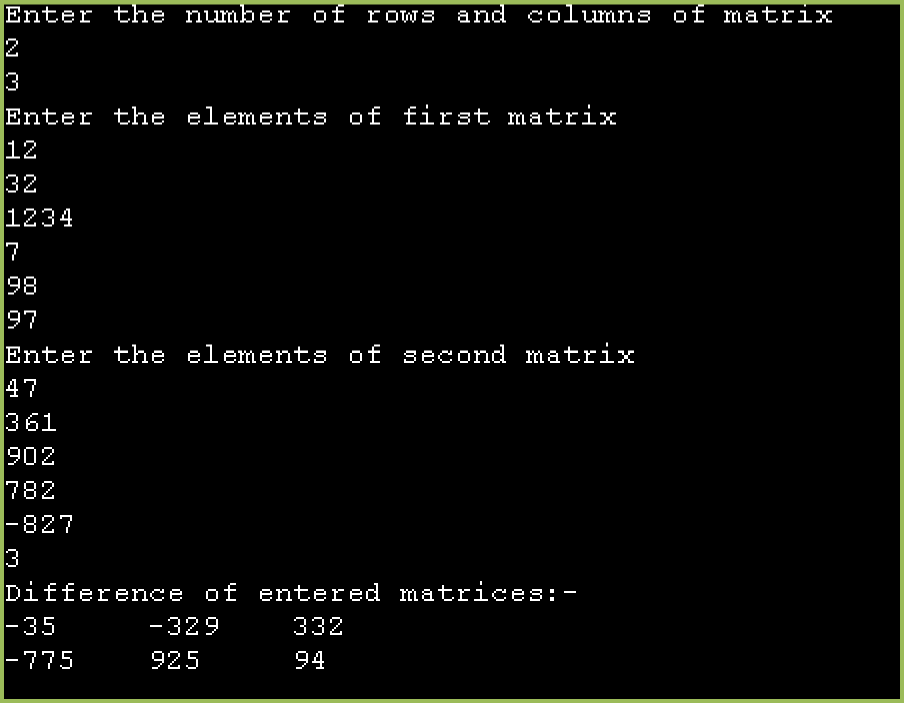
Source code:

#include<stdio.h>

```
int main()

{

   int m, n, c, d, first[10][10], second[10][10],
difference[10][10];

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

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

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

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

      for (d = 0; d < n; d++) scanf("%d", & first[c][d]);

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

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

      for (d = 0; d < n; d++) scanf("%d", & second[c][d]);

printf("Difference of entered matrices:-\n");

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

   {

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

      {

         difference[c][d] = first[c][d] - second[c][d];

printf("%d\t", difference[c][d]);
```

```
        }
        printf("\n");
    }
    return 0;
}
```

Output:



```
Enter the number of rows and columns of matrix
2
3
Enter the elements of first matrix
12
32
1234
7
98
97
Enter the elements of second matrix
47
361
902
782
-827
3
Difference of entered matrices:-
-35     -329    332
-775    925     94
```

21. Write a program in C for multiplication of two square Matrices.

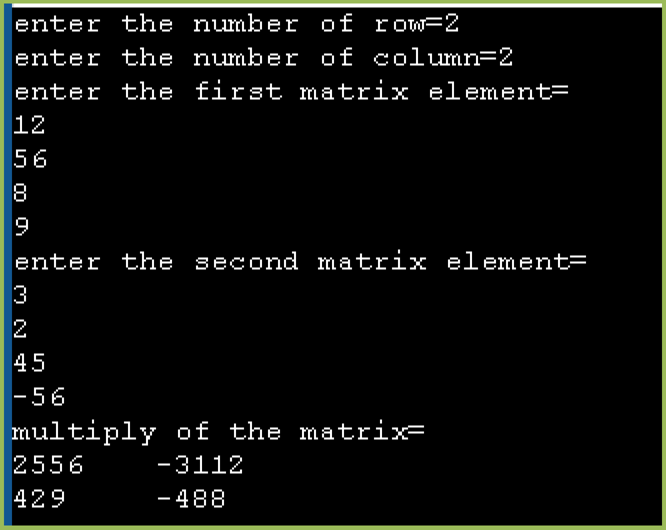
Ans

Source code:

```
#include<stdio.h>

#include<stdlib.h>

int main(){

int a[10][10],b[10][10],mul[10][10],r,c,i,j,k;


printf("enter the number of row=");

scanf("%d",&r);

printf("enter the number of column=");

scanf("%d",&c);

printf("enter the first matrix element=\n");

for(i=0;i<r;i++)

{

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

{

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

}

}

printf("enter the second matrix element=\n");
```

```
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
scanf("%d",&b[i][j]);
}
}

printf("multiply of the matrix=\n");
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
mul[i][j]=0;
for(k=0;k<c;k++)
{
mul[i][j]+=a[i][k]*b[k][j];
}
}
}
//for printing result
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
printf("%d\t",mul[i][j]);
}
printf("\n");
}
return 0;
}
```

Output:



```
enter the number of row=2
enter the number of column=2
enter the first matrix element=
12
56
8
9
enter the second matrix element=
3
2
45
-56
multiply of the matrix=
2556    -3112
429     -488
```

22. Write a program in C to find transpose of a given matrix.

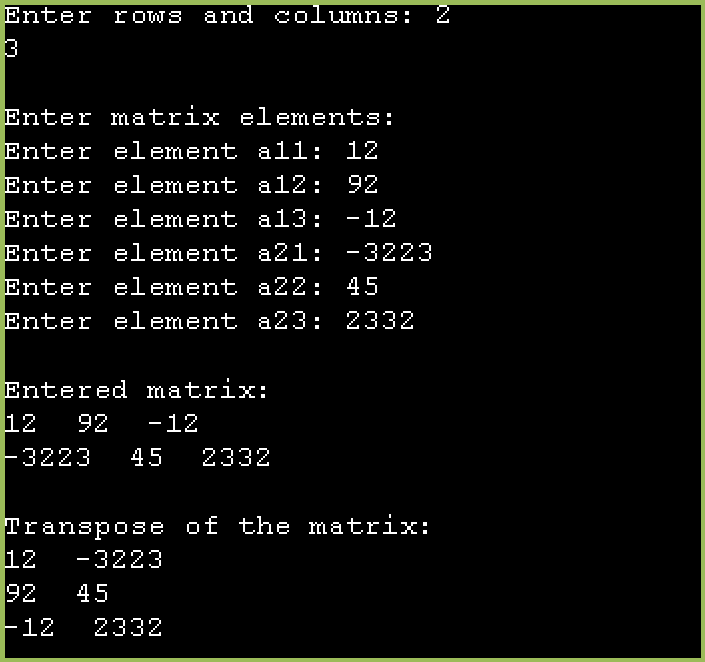
Ans

Source code:

```
#include <stdio.h>

int main() {

  int a[10][10], transpose[10][10], r, c;

printf("Enter rows and columns: ");

scanf("%d %d", &r, &c);


  // asssigning elements to the matrix

printf("\nEnter matrix elements:\n");
```

```
  for (int i = 0; i < r; ++i)
  for (int j = 0; j < c; ++j) {
printf("Enter element a%d%d: ", i + 1, j + 1);
scanf("%d", &a[i][j]);
  }

  // printing the matrix a[][]
printf("\nEntered matrix: \n");
  for (int i = 0; i < r; ++i)
  for (int j = 0; j < c; ++j) {
printf("%d  ", a[i][j]);
    if (j == c - 1)
    printf("\n");
  }

  // computing the transpose
  for (int i = 0; i < r; ++i)
  for (int j = 0; j < c; ++j) {
    transpose[j][i] = a[i][j];
  }

  // printing the transpose
printf("\nTranspose of the matrix:\n");
  for (int i = 0; i < c; ++i)
  for (int j = 0; j < r; ++j) {
printf("%d  ", transpose[i][j]);
    if (j == r - 1)
    printf("\n");
  }
  return 0;
}
```

Output:



```
Enter rows and columns: 2
3

Enter matrix elements:
Enter element a11: 12
Enter element a12: 92
Enter element a13: -12
Enter element a21: -3223
Enter element a22: 45
Enter element a23: 2332

Entered matrix:
12   92   -12
-3223   45   2332

Transpose of the matrix:
12   -3223
92   45
-12   2332
```

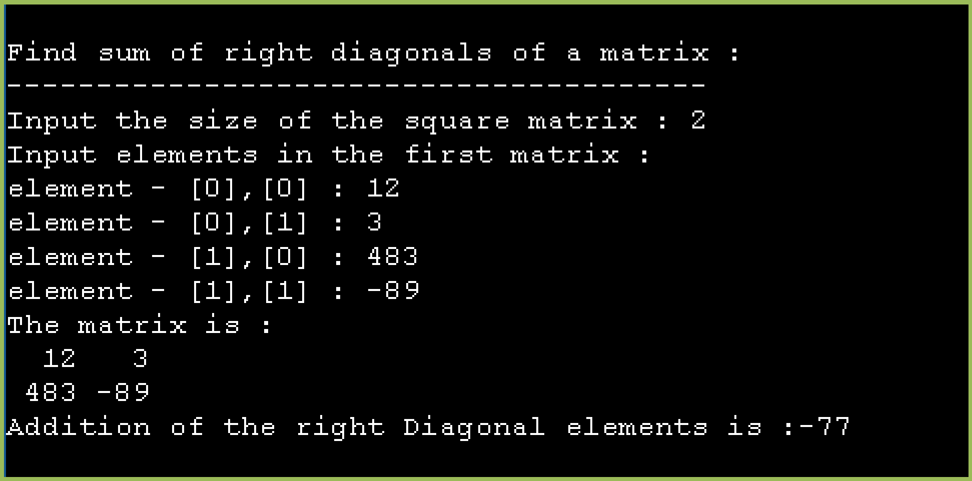
23. Write a program in C to find sum of right diagonals of a matrix.

Ans

```
#include <stdio.h>


void main()


  {

    int i,j,arr1[50][50],sum=0,n;


printf("\n\nFind sum of right diagonals of a matrix :\n");
```

```
        printf("--------------------------------------\n");


         printf("Input the size of the square matrix : ");
     scanf("%d", &n);
         printf("Input elements in the first matrix :\n");
        for(i=0;i<n;i++)
         {
           for(j=0;j<n;j++)
           {
         printf("element - [%d],[%d] : ",i,j);
                 scanf("%d",&arr1[i][j]);
                      if (i==j) sum= sum+arr1[i][j];
           }
         }



         printf("The matrix is :\n");
          for(i=0;i<n;i++)
          {
         for(j=0;j<n ;j++)
         printf("% 4d",arr1[i][j]);
            printf("\n");
          }

     printf("Addition of the right Diagonal elements is :%d\n",sum);
        }
```

Output:



```
Find sum of right diagonals of a matrix :
---------------------------------------
Input the size of the square matrix : 2
Input elements in the first matrix :
element - [0],[0] : 12
element - [0],[1] : 3
element - [1],[0] : 483
element - [1],[1] : -89
The matrix is :
  12   3
 483 -89
Addition of the right Diagonal elements is :-77
```

24. Write a program in C to find the sum of left diagonals of a matrix.

Ans

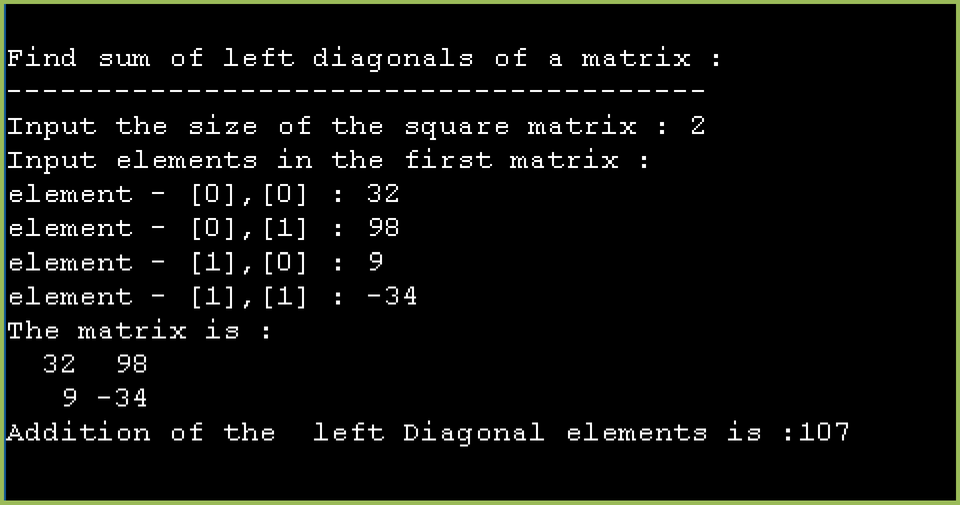
```
#include <stdio.h>
```

```
void main()

  {
    int i,j,arr1[50][50],sum=0,n,m=0;

printf("\n\nFind sum of left diagonals of a matrix :\n");
      printf("---------------------------------------\n");

        printf("Input the size of the square matrix : ");
scanf("%d", &n);
        m=n;
        printf("Input elements in the first matrix :\n");
      for(i=0;i<n;i++)
       {
          for(j=0;j<n;j++)
          {
        printf("element - [%d],[%d] : ",i,j);
```

```
                scanf("%d",&arr1[i][j]);
            }
        }
         printf("The matrix is :\n");
          for(i=0;i<n;i++)
          {
         for(j=0;j<n ;j++)
         printf("% 4d",arr1[i][j]);
             printf("\n");
          }
// calculate the sum of left diagonals
          for(i=0;i<n;i++)
          {
           m=m-1;
         for(j=0;j<n ;j++)
            {
              if (j==m)
                {
```

```
                sum= sum+arr1[i][j];
              }


          }
        }
printf("Addition of the  left Diagonal elements is :%d\n",sum);
   }
```

Output:



```
Find sum of left diagonals of a matrix :
---------------------------------------
Input the size of the square matrix : 2
Input elements in the first matrix :
element - [0],[0] : 32
element - [0],[1] : 98
element - [1],[0] : 9
element - [1],[1] : -34
The matrix is :
  32  98
   9 -34
Addition of the  left Diagonal elements is :107
```

25. Write a program in C to find sum of rows and columns of a Matrix.

Ans

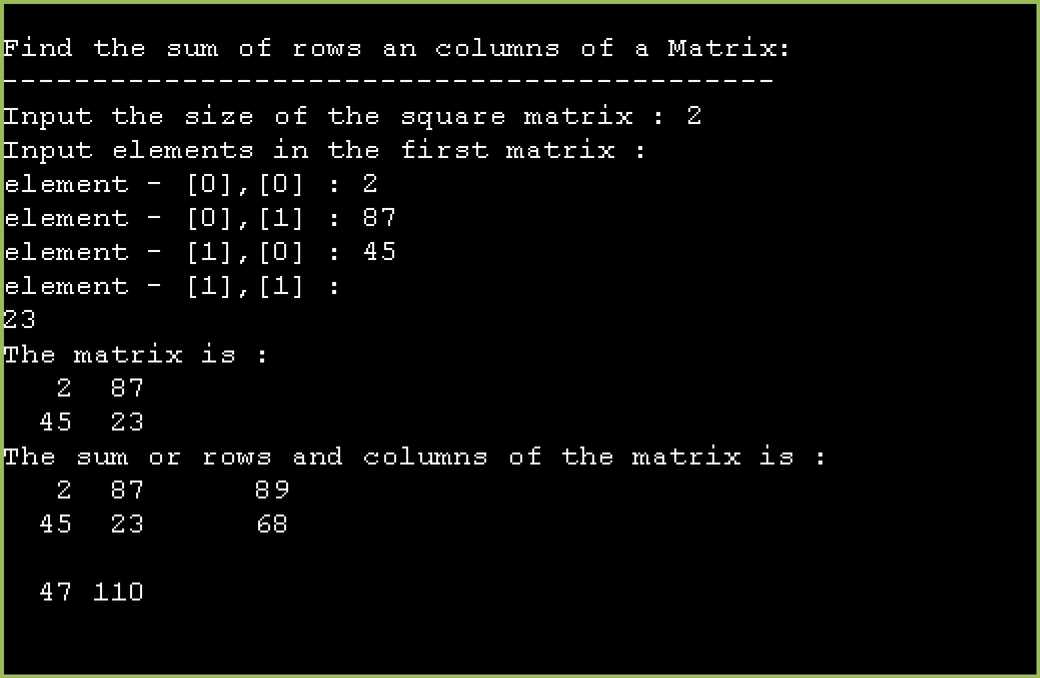
```
#include <stdio.h>
```

```
void main()

{

     int i,j,k,arr1[10][10],rsum[10],csum[10],n;


printf("\n\nFind the sum of rows an columns of a Matrix:\n");

       printf("-------------------------------------------\n");



printf("Input the size of the square matrix : ");

scanf("%d", &n);

         printf("Input elements in the first matrix :\n");

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

        {

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

            {

printf("element - [%d],[%d] : ",i,j);

                    scanf("%d",&arr1[i][j]);
            }
        }
         printf("The matrix is :\n");
          for(i=0;i<n;i++)
          {
         for(j=0;j<n ;j++)
         printf("% 4d",arr1[i][j]);
             printf("\n");
          }
     for(i=0;i<n;i++)
     {
          rsum[i]=0;
          for(j=0;j<n;j++)
          rsum[i]=rsum[i]+arr1[i][j];
     }
      for(i=0;i<n;i++)
      {
          csum[i]=0;
```

```
          for(j=0;j<n;j++)
                csum[i]=csum[i]+arr1[j][i];
     }

printf("The sum or rows and columns of the matrix is :\n");
     for(i=0;i<n;i++)
     {
          for(j=0;j<n;j++)
        printf("% 4d",arr1[i][j]);
        printf("% 8d",rsum[i]);
           printf("\n");
      }
      printf("\n");
           for(j=0;j<n;j++)
           {
        printf("% 4d",csum[j]);
           }
          printf("\n\n");
```

```
  }
```

Output:



```
Find the sum of rows an columns of a Matrix:
-------------------------------------------
Input the size of the square matrix : 2
Input elements in the first matrix :
element - [0],[0] : 2
element - [0],[1] : 87
element - [1],[0] : 45
element - [1],[1] :
23
The matrix is :
   2  87
  45  23
The sum or rows and columns of the matrix is :
   2  87       89
  45  23       68

  47 110
```

26. Write a program in C to print or display the lower triangular of a given matrix.

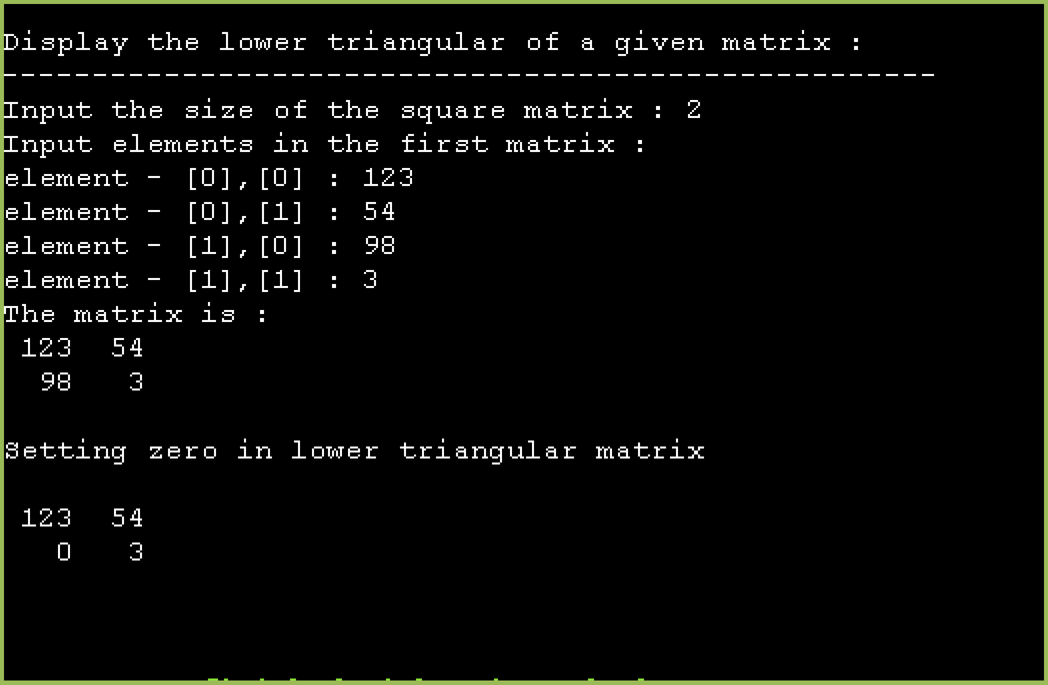
Ans

```
#include <stdio.h>


void main()

  {

 int arr1[10][10],i,j,n;
```

```
  float determinant=0;



printf("\n\nDisplay the lower triangular of a given matrix :\n");
       printf("--------------------------------------------------\n");



printf("Input the size of the square matrix : ");
scanf("%d", &n);
        printf("Input elements in the first matrix :\n");
       for(i=0;i<n;i++)
        {
            for(j=0;j<n;j++)
            {
        printf("element - [%d],[%d] : ",i,j);
                    scanf("%d",&arr1[i][j]);
            }
        }
        printf("The matrix is :\n");
         for(i=0;i<n;i++)
         {
        for(j=0;j<n ;j++)
        printf("% 4d",arr1[i][j]);
           printf("\n");
         }

printf("\nSetting zero in lower triangular matrix\n");
  for(i=0;i<n;i++){
    printf("\n");
    for(j=0;j<n;j++)
        if(i<=j)
printf("% 4d",arr1[i][j]);
        else
printf("% 4d",0);
  }
     printf("\n\n");
```

}

Output:



```
Display the lower triangular of a given matrix :
--------------------------------------------------
Input the size of the square matrix : 2
Input elements in the first matrix :
element - [0],[0] : 123
element - [0],[1] : 54
element - [1],[0] : 98
element - [1],[1] : 3
The matrix is :
 123  54
  98   3

Setting zero in lower triangular matrix

 123  54
   0   3
```

27. Write a program in C to print or display upper triangular matrix.

Ans

```
#include <stdio.h>
   void main()
   {
        int i, j, r, c, array[10][10];

printf("Enter the r and c value:");


scanf("%d%d", &r, &c);

        for (i = 1; i <= r; i++)

        {

            for (j = 1; j <= c; j++)

            {

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

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

            }

        }


printf("matrix is");

        for (i = 1; i <= r; i++)

        {

            for (j = 1; j <= c; j++)

            {
```

```
printf("%d", array[i][j]);
            }
            printf("\n");
        }

        for (i = 1; i <= r; i++)
        {
            printf("\n");
            for (j = 1; j <= c; j++)
            {
                if (i >= j)
                {
printf("%d", array[i][j]);
                }
                else
                {
                    printf("\t");
                }
            }
        }


        printf("\n\n");
        for (i = 1; i <= r; i++)
        {
            printf("\n");
            for (j = 1; j <= c; j++)
            {
                     if (j >= i)
                    {
         printf("%d", array[i][j]);
             }
             else
                    {
                     }
        }
    }
```

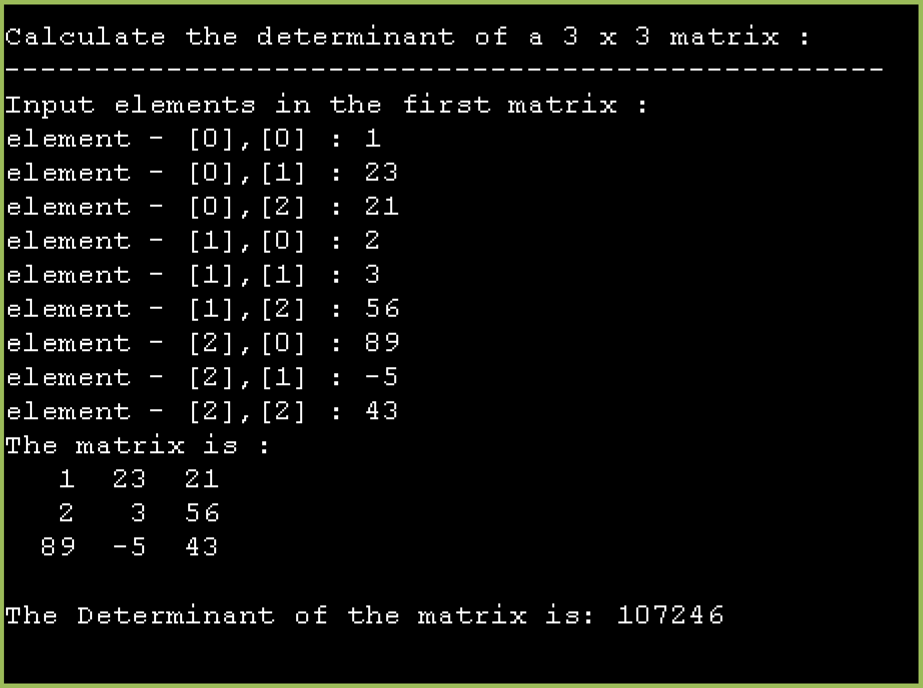
28. Write a program in C to calculate determinant of a 3 x 3 matrix.

Ans

```
#include <stdio.h>


void main()
  {
 int arr1[10][10],i,j,n;
 int det=0;



printf("\n\nCalculate the determinant of a 3 x 3 matrix :\n");
      printf("----------------------------------------------\n");


        printf("Input elements in the first matrix :\n");
      for(i=0;i<3;i++)
       {
          for(j=0;j<3;j++)
```

```
           {
        printf("element - [%d],[%d] : ",i,j);
                 scanf("%d",&arr1[i][j]);
           }
        }
        printf("The matrix is :\n");
         for(i=0;i<3;i++)
         {
        for(j=0;j<3 ;j++)
        printf("% 4d",arr1[i][j]);
            printf("\n");
         }
  for(i=0;i<3;i++)
      det = det + (arr1[0][i]*(arr1[1][(i+1)%3]*arr1[2][(i+2)%3] -
arr1[1][(i+2)%3]*arr1[2][(i+1)%3]));


printf("\nThe Determinant of the matrix is: %d\n\n",det);
}
```

Output:



```
Calculate the determinant of a 3 x 3 matrix :
-------------------------------------------------
Input elements in the first matrix :
element - [0],[0] : 1
element - [0],[1] : 23
element - [0],[2] : 21
element - [1],[0] : 2
element - [1],[1] : 3
element - [1],[2] : 56
element - [2],[0] : 89
element - [2],[1] : -5
element - [2],[2] : 43
The matrix is :
   1  23  21
   2   3  56
  89  -5  43

The Determinant of the matrix is: 107246
```

29. Write a program in C to accept a matrix and determine whether it is a sparse matrix.

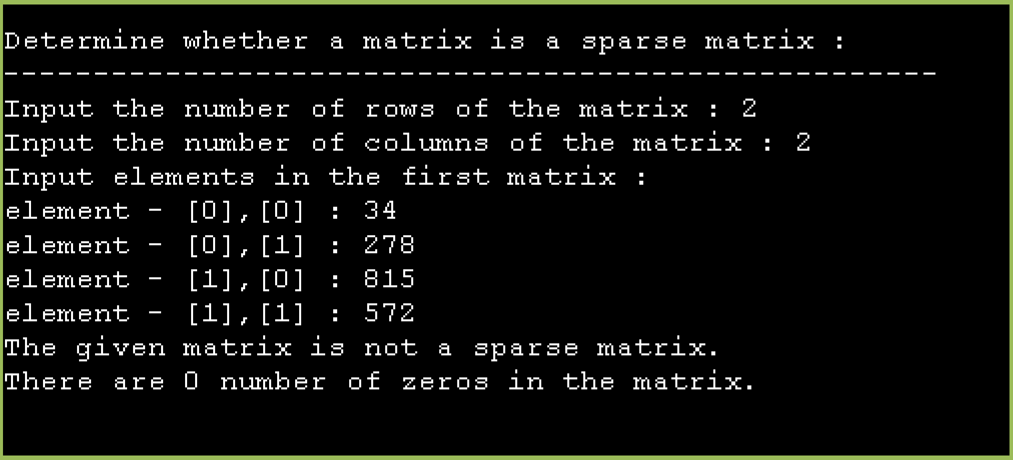
Ans

```
#include <stdio.h>

void main ()

{

        static int arr1[10][10];

        int i,j,r,c;

        int ctr=0;
```

```
printf("\n\nDetermine whether a matrix is a sparse matrix
:\n");
   printf("--------------------------------------------------\n");
printf("Input the number of rows of the matrix : ");
scanf("%d", &r);
printf("Input the number of columns of the matrix : ");
scanf("%d", &c);
      printf("Input elements in the first matrix :\n");
     for(i=0;i<r;i++)
      {
         for(j=0;j<c;j++)
         {
      printf("element - [%d],[%d] : ",i,j);
               scanf("%d",&arr1[i][j]);
                    if (arr1[i][j]==0)
                    {
                          ++ctr;
                    }
         }
```

```
        }
        if (ctr>((r*c)/2))
        {
                printf ("The given matrix is sparse matrix. \n");
        }
        else
                printf ("The given matrix is not a sparse matrix.\n");

        printf ("There are %d number of zeros in the
matrix.\n\n",ctr);
}
```

Output:



```
Determine whether a matrix is a sparse matrix :
----------------------------------------------------
Input the number of rows of the matrix : 2
Input the number of columns of the matrix : 2
Input elements in the first matrix :
element - [0],[0] : 34
element - [0],[1] : 278
element - [1],[0] : 815
element - [1],[1] : 572
The given matrix is not a sparse matrix.
There are 0 number of zeros in the matrix.
```

30. Write a program in C to accept two matrices and check whether they are equal.

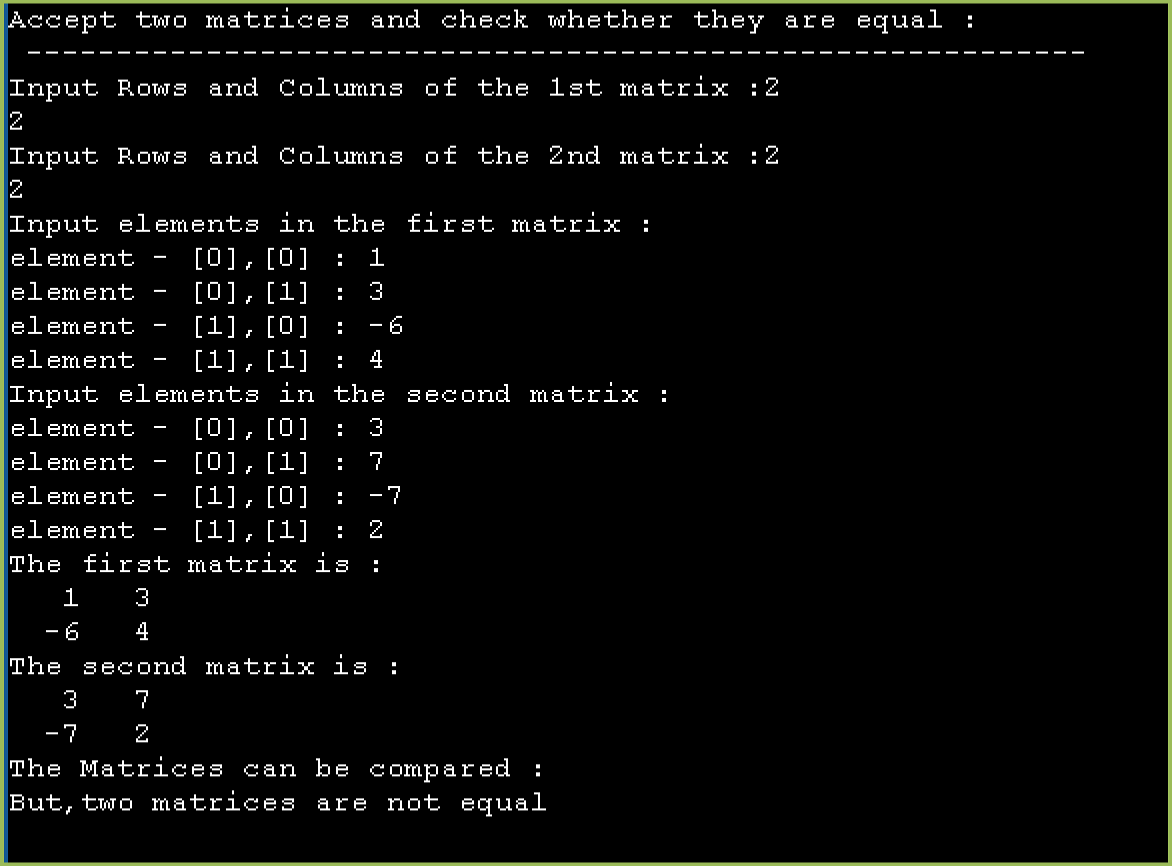
Ans

```
#include <stdio.h>

#include <stdlib.h>



void main()

{

  int arr1[50][50], brr1[50][50];

  int i, j, r1, c1, r2, c2, flag =1;


printf("\n\nAccept two matrices and check whether they are
equal :\n ");

      printf("-----------------------------------------------------\n");


printf("Input Rows and Columns of the 1st matrix :");

scanf("%d %d", &r1, &c1);


printf("Input Rows and Columns of the 2nd matrix :");

scanf("%d %d", &r2,&c2);
```

```
         printf("Input elements in the first matrix :\n");
        for(i=0;i<r1;i++)
         {
            for(j=0;j<c1;j++)
            {
         printf("element - [%d],[%d] : ",i,j);
                    scanf("%d",&arr1[i][j]);
            }
         }
     printf("Input elements in the second matrix :\n");
        for(i=0;i<r2;i++)
         {
            for(j=0;j<c2;j++)
            {
         printf("element - [%d],[%d] : ",i,j);
                    scanf("%d",&brr1[i][j]);
            }
         }
         printf("The first matrix is :\n");
          for(i=0;i<r1;i++)
          {
         for(j=0;j<c1 ;j++)
         printf("% 4d",arr1[i][j]);
             printf("\n");
          }
         printf("The second matrix is :\n");
          for(i=0;i<r2;i++)
          {
         for(j=0;j<c2 ;j++)
         printf("% 4d",brr1[i][j]);
             printf("\n");
          }



     if(r1 == r2 && c1 == c2)
        {
         printf("The Matrices can be compared : \n");
         for(i=0; i<r1; i++)
         {
             for(j=0; j<c2; j++)
             {
                 if(arr1[i][j] != brr1[i][j])
                 {
                     flag = 0;
                     break;
                 }
             }
          }
      }
      else
        {  printf("The Matrices Cannot be compared :\n");
        exit(1);
      }
        if(flag == 1 )
```

```
        printf("Two matrices are equal.\n\n");
    else
        printf("But,two matrices are not equal\n\n");
}
```

Output:



```
Accept two matrices and check whether they are equal :
 -----------------------------------------------------------
Input Rows and Columns of the 1st matrix :2
2
Input Rows and Columns of the 2nd matrix :2
2
Input elements in the first matrix :
element - [0],[0] : 1
element - [0],[1] : 3
element - [1],[0] : -6
element - [1],[1] : 4
Input elements in the second matrix :
element - [0],[0] : 3
element - [0],[1] : 7
element - [1],[0] : -7
element - [1],[1] : 2
The first matrix is :
   1   3
  -6   4
The second matrix is :
   3   7
  -7   2
The Matrices can be compared :
But,two matrices are not equal
```

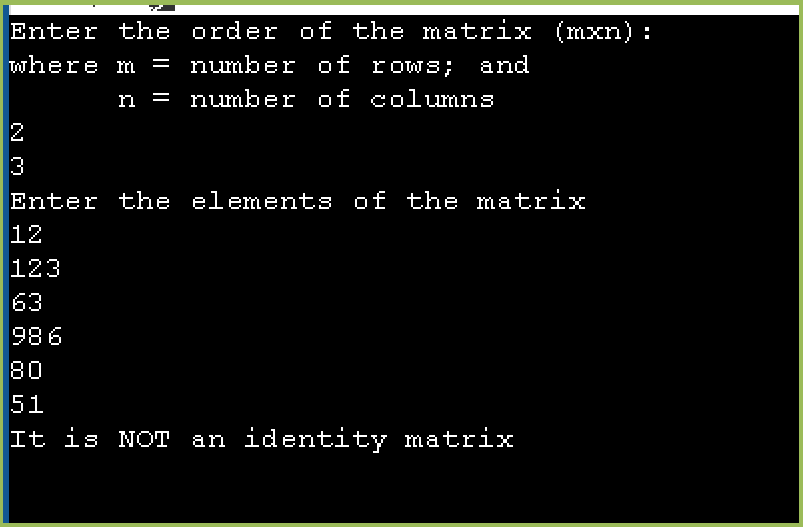
31. Write a program in C to check whether a given matrix is an identity matrix.

Ans

```c
#include <stdio.h>


int main (void)

{

        int a[10][10];

        int i = 0, j = 0, row = 0, col = 0;


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

        printf ("where m = number of rows; and\n");

        printf ("      n = number of columns\n");

        scanf ("%d %d", &row, &col);


        int flag = 0;


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

        for (i = 0; i < row; i++)

        {

                for (j = 0; j < col; j++)
        {
            scanf ("%d", &a[i][j]);
        }
    }

    for (i = 0; i < row; i++)
    {
        for (j = 0; j < col; j++)
        {
            if (i == j && a[i][j] != 1)
            {
                flag = -1;
                break;
            }
            else if (i != j && a[i][j] != 0)
            {
                flag = -1;
                break;
```

```
                    }
                }
            }

        if (flag == 0)
        {
            printf ("It is a IDENTITY MATRIX\n");
        }
        else
        {
            printf ("It is NOT an identity matrix\n");
        }

        return 0;
    }
```

Output:


```
Enter the order of the matrix (mxn):
where m = number of rows; and
      n = number of columns
2
3
Enter the elements of the matrix
12
123
63
986
80
51
It is NOT an identity matrix
```

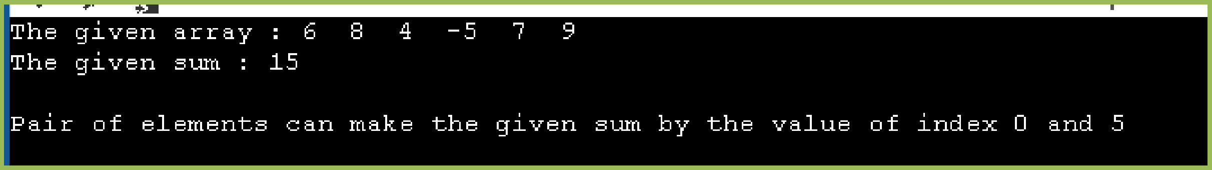
32. Write a program in C to find a pair with given sum in the array.

Ans

```
#include <stdio.h>

void checkForSum (int arr1[], int n, int s)

{



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

   {



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

       {
            if (arr1[i] + arr1[j] == s)
            {
printf("Pair of elements can make the given sum by the value of
index %d and %d", i, j);
                return;
            }
        }
    }
printf("No Pair can make the given sum.");
}

int main()
{
    int arr1[] = { 6, 8, 4, -5, 7, 9 };
    int s = 15;
printf("The given array : ");
    int n = sizeof(arr1)/sizeof(arr1[0]);
          for (int i = 0; i <= n - 1; i++)
```

```
   {
       printf("%d  ",arr1[i]);
       }
       printf("\nThe given sum : %d\n",s);
       printf("\n");
checkForSum(arr1, n, s);
   return 0;
}
```

Output:



```
The given array : 6   8   4   -5   7   9
The given sum : 15

Pair of elements can make the given sum by the value of index 0 and 5
```

33.Write a program in C to find the majority element of an array. A majority element in an array A[] of size n is an element that appears more than n/2 times (and hence there is at most one such element).

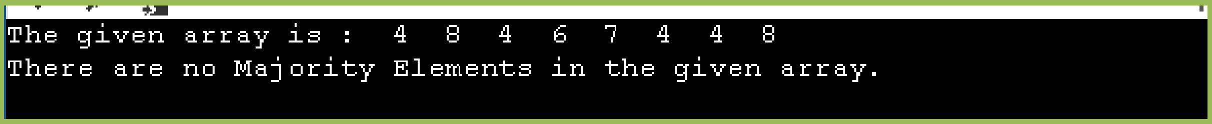
Ans

```
#include <stdio.h>


void findMajElem(int *arr1, int n)
```

```
{
 int i,IndexOfMajElem = 0, ctr = 1;
for(i = 1; i < n; i++)
        {
        if(arr1[IndexOfMajElem] == arr1[i])
            ctr++;
        else
            ctr--;

if(ctr == 0) {
            IndexOfMajElem = i;
            ctr = 1;
        }
    }
    ctr = 0;
    for (i = 0; i < n; i++)
        {
        if(arr1[i] == arr1[IndexOfMajElem])
```

```
            ctr++;
     }
    if(ctr > (n/2))
    printf("Majority Element : %d\n", arr1[IndexOfMajElem]);
        else
    printf("There are no Majority Elements in the given array.\n");
    }

    int main()
    {
        int i, ctr,m;
        int arr1[] = { 4, 8, 4, 6, 7, 4 , 4, 8};
        ctr = sizeof(arr1)/sizeof(arr1[0]);
    printf("The given array is :  ");

            for(i = 0; i < ctr; i++)
            {
            printf("%d  ", arr1[i]);
        }

        printf("\n");

            findMajElem(arr1, ctr);

        return 0;

    }
```

Output:



```
The given array is :   4   8   4   6   7   4   4   8
There are no Majority Elements in the given array.
```

34. Write a program in C to find the number occurring odd number of times in an array.

Ans

```
#include <stdio.h>


int findOddCountElem (int *arr1, int n )

{

    int i, ResultXor = 0;

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

        {
```

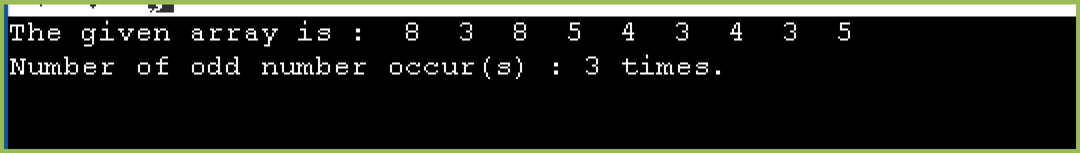
```
      ResultXor = ResultXor ^ arr1[i];
         }
     return ResultXor;
}

int main()
{
    int i;
 int arr1[] = {8, 3, 8, 5, 4, 3, 4, 3, 5};

    int ctr = sizeof(arr1)/sizeof(arr1[0]);
printf("The given array is :  ");

         for(i = 0; i < ctr; i++)
         {
         printf("%d  ", arr1[i]);
    }
    printf("\n");
```

printf("Number of odd number occur(s) : %d times.\n", findOddCountElem(arr1, ctr));

 return 0;

}

Output:



```
The given array is :  8  3  8  5  4  3  4  3  5
Number of odd number occur(s) : 3 times.
```

35. Write a program in C to find the largest sum of contiguous subarray of an array.

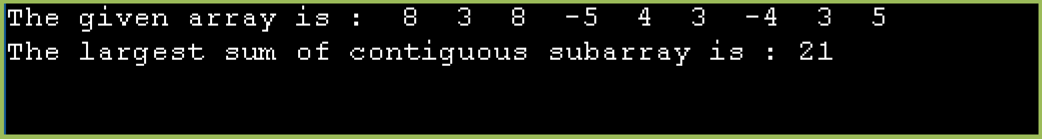
Ans

```
#include <stdio.h>


int maxSum(int a[],int n)

{

  int i,j,k;

  int sum,maxSum = 0;

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

  {
```

```
for(j=i; j<n; j++)
    {
      sum = 0;
for(k=i ; k<j; k++)
      {
        sum = sum + a[k];
      }
      if(sum>maxSum)
        maxSum = sum;
    }
   }
   return maxSum;
}

int main()
{
    int i;
 int arr1[] = {8, 3, 8, -5, 4, 3, -4, 3, 5};
```

```
    int ctr = sizeof(arr1)/sizeof(arr1[0]);
printf("The given array is :  ");


        for(i = 0; i < ctr; i++)
        {
        printf("%d  ", arr1[i]);
    }
    printf("\n");


printf("The largest sum of contiguous subarray is : %d \n", maxSum(arr1, ctr));
 return 0;
}
```

Output:



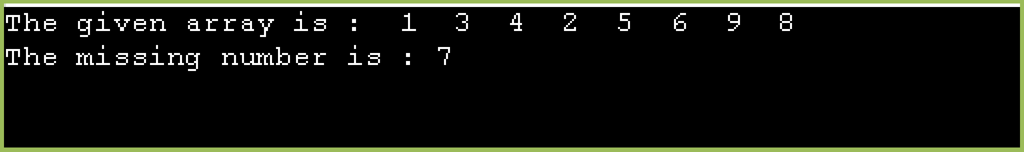
```
The given array is :   8   3   8   -5   4   3   -4   3   5
The largest sum of contiguous subarray is : 21
```

36. Write a program in C to find the missing number from a given array. There are no duplicates in list.

Ans

```
#include <stdio.h>
 int pickMissNumber(int *arr1, int ar_size)
{
    int i, sum = 0, n = ar_size + 1;
for(i = 0; i < ar_size; i++)
         {
        sum = sum + arr1[i];
    }
    return (n*(n+1))/2 - sum;
}
int main()
{
    int i;
    int arr1[] = {1, 3, 4, 2, 5, 6, 9, 8};

    int ctr = sizeof(arr1)/sizeof(arr1[0]);
printf("The given array is :  ");


        for(i = 0; i < ctr; i++)

        {

        printf("%d  ", arr1[i]);

    }

    printf("\n");


printf("The missing number is : %d \n", pickMissNumber(arr1,
ctr));

 return 0;

}
```

Output:



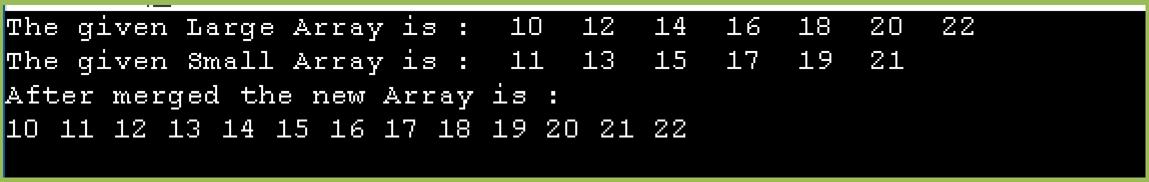
```
The given array is :   1   3   4   2   5   6   9   8
The missing number is : 7
```

38. Write a program in C to merge one sorted array into another sorted array.

Ans

```
#include <stdio.h>


void merge2arrs(int *bgArr, int bgArrCtr, int *smlArr, int
smlArrCtr)
{
if(bgArr == NULL || smlArr == NULL)
        return;
    int bgArrIndex = bgArrCtr-1,
    smlArrIndex = smlArrCtr-1,
    mergedArrayIndex = bgArrCtr + smlArrCtr -1;
while(bgArrIndex >= 0 && smlArrIndex >= 0) {
 if(bgArr[bgArrIndex] >= smlArr[smlArrIndex]){
            bgArr[mergedArrayIndex] = bgArr[bgArrIndex];
            mergedArrayIndex--;
            bgArrIndex--;
        } else {
            bgArr[mergedArrayIndex] = smlArr[smlArrIndex];
            mergedArrayIndex--;
            smlArrIndex--;
        }
    }
if(bgArrIndex < 0)
        {
while(smlArrIndex >= 0)
                {
            bgArr[mergedArrayIndex] = smlArr[smlArrIndex];
            mergedArrayIndex--;
            smlArrIndex--;
  }
    } else if (smlArrIndex < 0)
        {
while(bgArrIndex >= 0)
                {
            bgArr[mergedArrayIndex] = bgArr[bgArrIndex];
            mergedArrayIndex--;
            bgArrIndex--;
        }
    }
}

int main()
{
    int bigArr[13] = {10, 12, 14, 16, 18, 20, 22};
    int smlArr[6] = {11, 13, 15, 17, 19, 21};
    int i;

printf("The given Large Array is :  ");
        for(i = 0; i < 7; i++)
        {
                printf("%d  ", bigArr[i]);
    }
    printf("\n");

printf("The given Small Array is :  ");
        for(i = 0; i < 6; i++)
        {
                printf("%d  ", smlArr[i]);
    }
    printf("\n");

    merge2arrs(bigArr, 7, smlArr, 6);

printf("After merged the new Array is :\n");
for(i = 0; i<13; i++)
        {
                printf("%d ", bigArr[i]);
    }
    return 0;
}
```

Output:



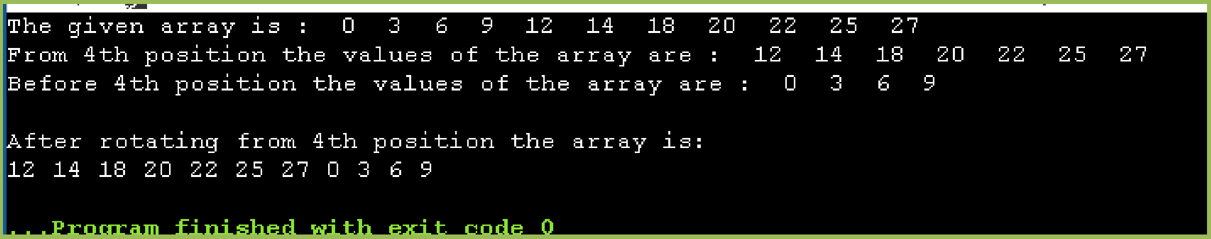
```
The given Large Array is :  10   12   14   16   18   20   22
The given Small Array is :  11   13   15   17   19   21
After merged the new Array is :
10 11 12 13 14 15 16 17 18 19 20 21 22
```

39. Write a program in C to rotate an array by N positions.

Ans

```
#include <stdio.h>

void shiftArr1Pos(int *arr1, int arrSize)

{

    int i, temp;

      temp = arr1[0];

for(i = 0; i < arrSize-1; i++)

        {

        arr1[i] = arr1[i+1];

    }

    arr1[i] = temp;

}

void arr1Rotate(int *arr1, int arrSize, int rotFrom)

{

    int i;

for(i = 0; i < rotFrom; i++)

        {

        shiftArr1Pos(arr1, arrSize);
    }
    return;
}
int main()
{
    int arr1[] = {0,3,6,9,12,14,18,20,22,25,27};
        int ctr = sizeof(arr1)/sizeof(arr1[0]);
    int i;

        printf("The given array is :  ");
        for(i = 0; i < ctr; i++)
        {
        printf("%d  ", arr1[i]);
    }
    printf("\n");

        printf("From 4th position the values of the array are :  ");
        for(i = 4; i < ctr; i++)
        {
        printf("%d  ", arr1[i]);
    }
    printf("\n");

        printf("Before 4th position the values of the array are :  ");
        for(i = 0; i < 4; i++)
        {
        printf("%d  ", arr1[i]);
    }
    printf("\n");

    arr1Rotate(arr1, ctr, 4);
printf("\nAfter rotating from 4th position the array is: \n");
for(i = 0; i<ctr; i++)
        {
printf("%d ", arr1[i]);
    }
    return 0;

}
```

Output:



```
The given array is :  0  3  6  9  12  14  18  20  22  25  27
From 4th position the values of the array are :  12  14  18  20  22  25  27
Before 4th position the values of the array are :  0  3  6  9

After rotating from 4th position the array is:
12 14 18 20 22 25 27 0 3 6 9

...Program finished with exit code 0
```

43. Write a program in C to to print next greater elements in a given unsorted array. Elements for which no greater element exist, consider next greater element as -1.

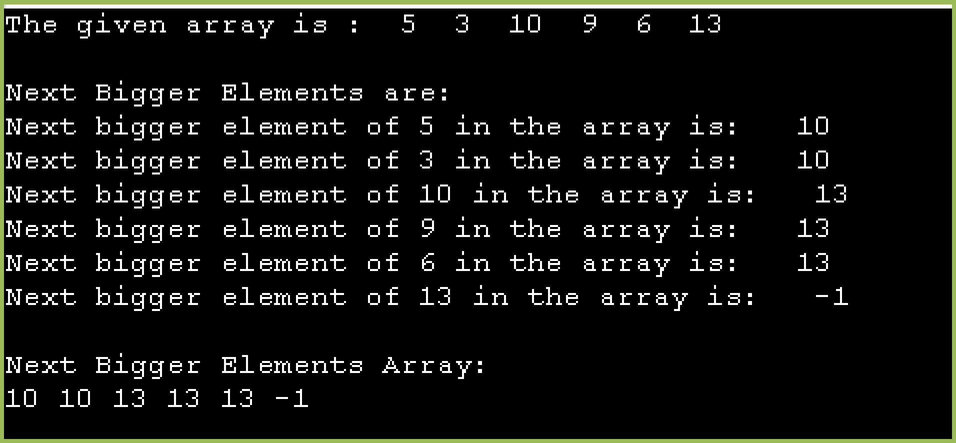
Ans

```
#include <stdio.h>


void findNxtLrgElem (int *arr1, int arr1_size)

{

    int nxtBgElem, i, j;

for(i = 0; i < arr1_size; i++)

        {

       for (j = i+1, nxtBgElem = -1; j < arr1_size; j++)
```

```
                 {
            if (arr1[i] < arr1[j])
                          {
                nxtBgElem = arr1[j];
                break;
            }
        }
printf("Next bigger element of %d in the array is:   %d\n",
arr1[i], nxtBgElem);
    }
}
void formBigElemArray (int *arr1, int arr1_size)
{
    int nxtBgElem, i, j;
for(i = 0; i < arr1_size; i++)
         {
        for (j = i+1, nxtBgElem = -1; j < arr1_size; j++)
                 {
            if (arr1[i] < arr1[j])
```

```
                        {
                nxtBgElem = arr1[j];
                break;
            }
        }
printf("%d ", nxtBgElem);
    }
}
int main()
{
    int i, arr1[]= {5, 3, 10, 9, 6, 13};
    int ctr = sizeof(arr1) / sizeof(arr1[0]);

        printf("The given array is :  ");
        for(i = 0; i < ctr; i++)
        {
        printf("%d  ", arr1[i]);
    }
    printf("\n");


printf("\nNext Bigger Elements are:\n");

findNxtLrgElem(arr1, ctr);


printf("\nNext Bigger Elements Array:\n");

formBigElemArray(arr1, ctr);

    return 0;

}
```

Output:



```
The given array is :  5  3  10  9  6  13

Next Bigger Elements are:
Next bigger element of 5 in the array is:   10
Next bigger element of 3 in the array is:   10
Next bigger element of 10 in the array is:   13
Next bigger element of 9 in the array is:   13
Next bigger element of 6 in the array is:   13
Next bigger element of 13 in the array is:   -1

Next Bigger Elements Array:
10 10 13 13 13 -1
```

45. Write a program in C to find two elements whose sum is closest to zero.

Ans

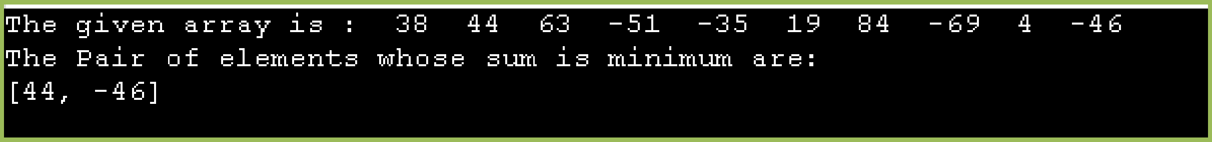
```
#include <stdio.h>
```

```
#include <math.h>

#include <stdlib.h>



void findMinSumPair(int *arr1, int arr_size)

{

  int i, j, sum, minSum, min1Pair, min2Pair;


if(arr1 == NULL || arr_size < 2)

      return;

  min1Pair = arr1[0];

  min2Pair = arr1[1];

  minSum = min1Pair + min2Pair;


for(i = 0; i < arr_size-1; i++)

  {

for(j = i+1; j < arr_size; j++)

        {
```

```
      sum = arr1[i] + arr1[j];
      if(abs(sum) < abs(minSum))
          {
        minSum = sum;
        min1Pair = arr1[i];
        min2Pair = arr1[j];
      }
    }
  }
printf("[%d, %d]\n", min1Pair, min2Pair);
}

int main()
{
    int arr1[] = {38, 44, 63, -51, -35, 19, 84, -69, 4, -46};
  int ctr = sizeof(arr1)/sizeof(arr1[0]);
  int i;
```

```
        printf("The given array is :  ");
        for(i = 0; i < ctr; i++)
        {
        printf("%d  ", arr1[i]);
    }
    printf("\n");

printf("The Pair of elements whose sum is minimum are: \n");
findMinSumPair(arr1, ctr);
    return 0;
}
```

Output:



```
The given array is :   38   44   63   -51   -35   19   84   -69   4   -46
The Pair of elements whose sum is minimum are:
[44, -46]
```

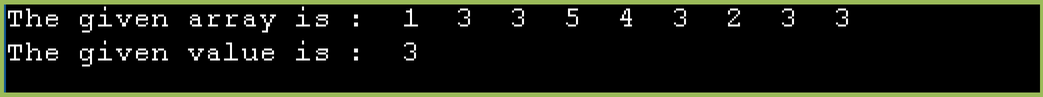
48. Write a program in C to find if a given integer x appears more than n/2 times in a sorted array of n integers.

Ans

```
# include <stdio.h>
# include <stdbool.h>

bool ChkMajority(int arr1[], int arr_size, int x)
{
    int i;
    int last_index = arr_size%2? (arr_size/2+1): (arr_size/2);
    {
        if (arr1[i] == x && arr1[i+arr_size/2] == x)
            return 1;
    }
    return 0;
}
int main()
{
     int arr1[] ={1, 3, 3, 5, 4, 3, 2, 3, 3};
     int arr_size = sizeof(arr1)/sizeof(arr1[0]);
     int x = 3,i;
```

```
        printf("The given array is :  ");
        for(i = 0; i < arr_size; i++)
        {
        printf("%d  ", arr1[i]);
    }
        printf("\n");
        printf("The given value is :  %d\n",x);



     if (ChkMajority(arr1, arr_size, x))
printf("%d appears more than %d times in the given array[]",x,
arr_size/2);
     else
printf("%d does not appear more than %d times in the given
array[]", x, arr_size/2);


   return 0;
}
```

Output:



```
The given array is :  1  3  3  5  4  3  2  3  3
The given value is :  3
```

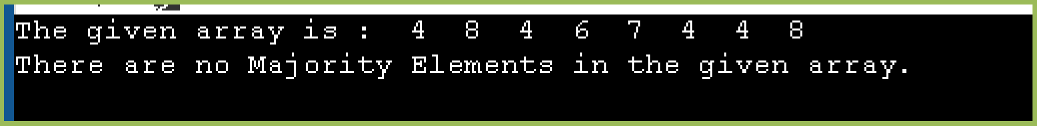
49. Write a program in C to find majority element of an array.

Ans

```
#include <stdio.h>


void findMajElem(int *arr1, int n)

{

 int i,IndexOfMajElem = 0, ctr = 1;

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

        {

        if(arr1[IndexOfMajElem] == arr1[i])

            ctr++;

        else

            ctr--;


if(ctr == 0) {

            IndexOfMajElem = i;
            ctr = 1;
        }
    }
    ctr = 0;
    for (i = 0; i < n; i++)
        {
        if(arr1[i] == arr1[IndexOfMajElem])
            ctr++;
 }
if(ctr > (n/2))
printf("Majority Element : %d\n", arr1[IndexOfMajElem]);
    else
printf("There are no Majority Elements in the given array.\n");
}

int main()
{
    int i, ctr,m;
```

```
    int arr1[] = { 4, 8, 4, 6, 7, 4 , 4, 8};

    ctr = sizeof(arr1)/sizeof(arr1[0]);

printf("The given array is :  ");


        for(i = 0; i < ctr; i++)

        {

        printf("%d  ", arr1[i]);

    }

    printf("\n");

        findMajElem(arr1, ctr);

    return 0;

}
```

Output:



```
The given array is :  4  8  4  6  7  4  4  8
There are no Majority Elements in the given array.
```

55. Write a program in C to check whether an array is subset of another array.

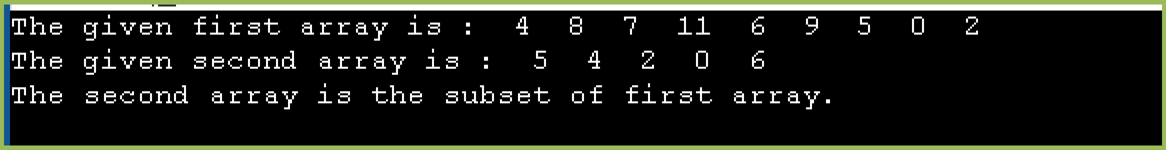
Ans

```
#include <stdio.h>
```

```
int chkSubsetArray(int *arr1 , int arr1_size, int *arr2, int
arr2_size)
{
  int i, j;
  for (i = 0; i < arr2_size; i++)
     {
     for (j = 0; j < arr1_size; j++)
           {
       if(arr2[i] == arr1[j])
         break;
     }
if(j == arr1_size)
       return 0;
  }
  return 1;
}
int main()
{
  int arr1[] = {4, 8, 7, 11, 6, 9, 5, 0, 2};
  int arr2[] = {5, 4, 2, 0, 6};

  int n1 = sizeof(arr1)/sizeof(arr1[0]);

     int i;

  int n2 = sizeof(arr2)/sizeof(arr2[0]);


     printf("The given first array is :  ");

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

     {

     printf("%d  ", arr1[i]);

  }

     printf("\n");


     printf("The given second array is :  ");

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

     {

     printf("%d  ", arr2[i]);

  }

     printf("\n");
```

```
if(chkSubsetArray(arr1, 9, arr2, 4))
printf("The second array is the subset of first array.");
   else
printf("The second array is not a subset of first array");


   return 0;
}
```

Output:



```
The given first array is :   4   8   7   11   6   9   5   0   2
The given second array is :   5   4   2   0   6
The second array is the subset of first array.
```

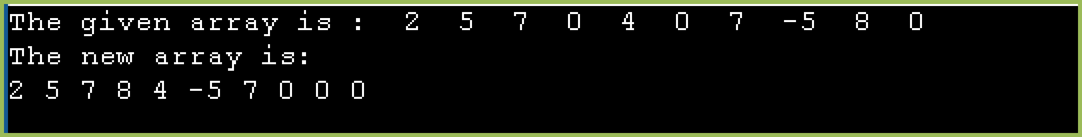
58. Write a program in C to move all zeroes to the end of a given array.

Ans

```
#include <stdio.h>


void PickOutZeros (int *arr1, int arr_size)
{
    int tmp, lft = 0, rgt = arr_size-1;
while(rgt > lft)
        {
     while(arr1[lft] != 0)
         lft++;
     while(arr1[rgt] == 0)
         rgt--;
if(lft < rgt)
         {
            tmp = arr1[lft];
            arr1[lft] = arr1[rgt];
            arr1[rgt] = tmp;
        }
    }
}

int main()
{
    int arr1[] = {2, 5, 7, 0, 4, 0, 7, -5, 8, 0};

   int n = sizeof(arr1)/sizeof(arr1[0]);

   int i;


        printf("The given array is :  ");

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

        {

        printf("%d  ", arr1[i]);

    }

        printf("\n");


PickOutZeros(arr1, n);

        printf("The new array is: \n");

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

        {

printf("%d ", arr1[i]);

    }

    return 0;
```

}

Output:



```
The given array is :   2   5   7   0   4   0   7   -5   8   0
The new array is:
2 5 7 8 4 -5 7 0 0 0
```

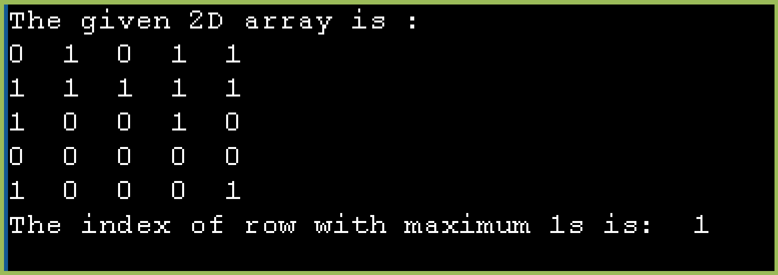
60. Write a program in C to find the row with maximum number of 1s.

Ans

```
#include <stdio.h>

#define R 5

#define C 5




int getFirstOccur(int arr1[], int l, int h)

{

if(h >= l)

{

   int mid = l + (h - l)/2;

   if ( ( mid == 0 || arr1[mid-1] == 0) && arr1[mid] == 1)
```

```
    return mid;

    else if (arr1[mid] == 0)

    return getFirstOccur(arr1, (mid + 1), h);

    else

    return getFirstOccur(arr1, l, (mid -1));

}

return -1;

}

int findRowMaxOne(int arr2d[R][C])

{

    int max_row_index = 0, max = -1;

    int i, index;

    for (i = 0; i < R; i++)

    {

    index = getFirstOccur (arr2d[i], 0, C-1);

    if (index != -1 && C-index > max)

    {

        max = C - index;
        max_row_index = i;
    }
    }
    return max_row_index;
}

int main()
{
    int arr2d[R][C] = { {0, 1, 0, 1,1},
                       {1, 1, 1, 1, 1},
                       {1, 0, 0, 1, 0},
                       {0, 0, 0, 0, 0},
                       {1, 0, 0, 0, 1}
    };
    int i,j;

        printf("The given 2D array is :  \n");
        for(i = 0; i < R; i++)
        {
        for(j=0; j<C ; j++)
        {
        printf("%d  ", arr2d[i][j]);
        }

        printf("\n");
    }

printf("The index of row with maximum 1s is:  %d " ,
findRowMaxOne(arr2d));
    return 0;
}
```

Output:



```
The given 2D array is :
0   1   0   1   1
1   1   1   1   1
1   0   0   1   0
0   0   0   0   0
1   0   0   0   1
The index of row with maximum 1s is:   1
```

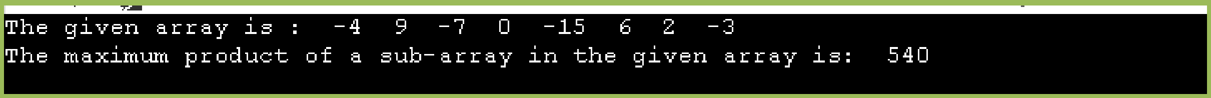
61. Write a program in C to find maximum product subarray in a given array.

Ans

```
#include <stdio.h>

int min(int p, int q)

{

    return (p < q) ? p : q;

}

int max(int p, int q)

{

    return (p > q) ? p : q;

}

int maxProduct(int arr1 [], int n)

{

    int maxend = 0, minend = 0;

    int maxupto = 0;

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

    {

        int temp = maxend;
        maxend = max(arr1[i], max(arr1[i] * maxend, arr1[i] *
minend));

        minend = min(arr1[i], min(arr1[i] * temp, arr1[i] *
minend));

        maxupto = max(maxupto, maxend);

    }

    return maxupto;

}

int main(void)

{

    int arr1[] = { -4, 9, -7, 0, -15, 6, 2, -3 };

    int n = sizeof(arr1) / sizeof(arr1[0]);

    int i;


        printf("The given array is :  ");

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

        {

        printf("%d  ", arr1[i]);

    }
```

```
        printf("\n");

printf("The maximum product of a sub-array in the given array is:  %d", maxProduct(arr1, n));
    return 0;
}
```

Output:



```
The given array is :  -4  9  -7  0  -15  6  2  -3
The maximum product of a sub-array in the given array is:  540
```

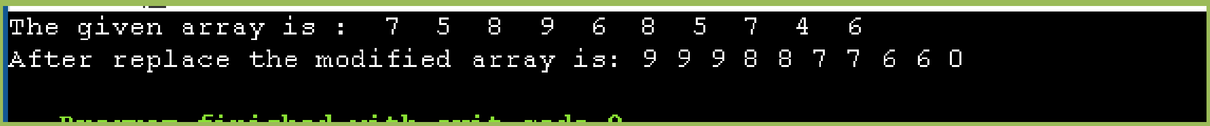
63. Write a program in C to replace every element with the greatest element on its right side.

Source code:

```
#include <stdio.h>


void printArray(int a[] ,int n)
{
for(int i = 0;i < n;i++)
printf("%d ",a[i]);
}
```

```
void replaceWithNextGreatest(int a[], int size)
{
   int maximum =  a[size-1];
   a[size-1] = 0;
for(int i = size-2; i >= 0; i--)
   {
      int temp = a[i];
      a[i] = maximum;
if(maximum < temp)
         maximum = temp;
   }
printf("After replace the modified array is: ");
printArray(a , size);
}

int main()
{
   int i, arr1[] = {7, 5, 8, 9, 6, 8, 5, 7, 4, 6};

   int n = sizeof(arr1) / sizeof(arr1[0]);


       printf("The given array is :  ");

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

       {

       printf("%d  ", arr1[i]);

   }

       printf("\n");


replaceWithNextGreatest(arr1, n);

   return 0;

}
```

Output:



```
The given array is :   7   5   8   9   6   8   5   7   4   6
After replace the modified array is: 9 9 9 8 8 7 7 6 6 0

  ...Program finished with exit code 0
```

64. Write a program in C to find the median of two sorted arrays of same size.

Ans

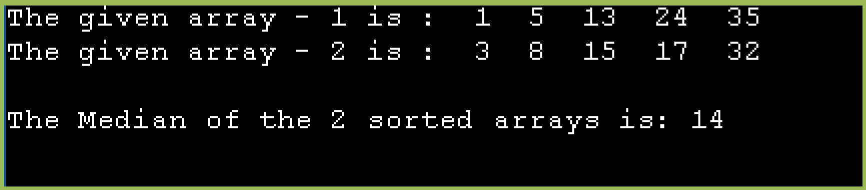
```
#include <stdio.h>


int max(int a, int b)

{

   return ((a > b) ? a : b);

}

int min(int a, int b)

{

   return ((a < b) ? a : b);

}

int median(int arr[], int size)

{

   if (size % 2 == 0)

         return (arr[size/2] + arr[size/2-1])/2;

   else

         return arr[size/2];

}

int median2SortedArrays(int arr1[], int arr2[], int size)
```

```
{
   int med1;
   int med2;
if(size <= 0) return -1;
if(size == 1) return (arr1[0] + arr2[0])/2;
   if (size == 2) return (max(arr1[0], arr2[0]) + min(arr1[1],
arr2[1])) / 2;

   med1 = median(arr1, size);
   med2 = median(arr2, size);

if(med1 == med2) return med1;

   if (med1 < med2)
   {
      return median2SortedArrays(arr1 + size/2, arr2, size -
size/2);
   }
   else
```

```
  {
    return median2SortedArrays(arr2 + size/2, arr1, size -
size/2);
  }
}

int main()
{
  int i,m,n;
  int arr1[] = {1, 5, 13, 24, 35};
  int arr2[] = {3, 8, 15, 17, 32};
  m = sizeof(arr1) / sizeof(arr1[0]);
  n = sizeof(arr2) / sizeof(arr2[0]);

      printf("The given array - 1 is :  ");
      for(i = 0; i < m; i++)
      {
      printf("%d  ", arr1[i]);
   }
```

```
        printf("\n");


        printf("The given array - 2 is :  ");
        for(i = 0; i < n; i++)
        {
        printf("%d  ", arr2[i]);
    }
        printf("\n");


printf("\nThe Median of the 2 sorted arrays is:
%d",median2SortedArrays(arr1, arr2, n));
  printf("\n");
  return 0;
}
```

Output:



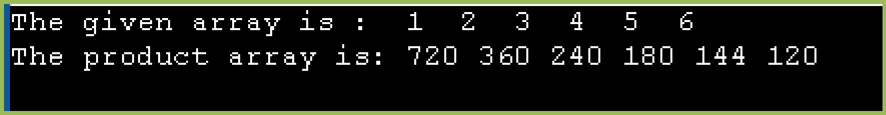
```
The given array - 1 is :   1   5   13   24   35
The given array - 2 is :   3   8   15   17   32

The Median of the 2 sorted arrays is: 14
```

65. Write a program in C to find the product of an array such that product is equal to the product of all the elements of arr[] except arr[i].

Ans

```
#include <stdio.h>


void productOfArray(int arr1[], int n)
{
  int l_arr[n],r_arr[n],product[n];
  int i, j;
  l_arr[0] = 1;
  r_arr[n-1] = 1;
for(i = 1; i < n; i++)
    l_arr[i] = arr1[i-1]*l_arr[i-1];
for(j = n-2; j >=0; j--)
    r_arr[j] = arr1[j+1]*r_arr[j+1];
  for (i=0; i<n; i++)
    product[i] = l_arr[i] * r_arr[i];
  for (i=0; i<n; i++)
printf("%d ",product[i]);
}

int main()
{
  int arr1[] = {1, 2, 3, 4, 5, 6};
  int n = sizeof(arr1)/sizeof(arr1[0]);
  int i;

    printf("The given array is :  ");
    for(i = 0; i < n; i++)
    {
    printf("%d  ", arr1[i]);
     }
    printf("\n");

printf("The product array is: ");
productOfArray(arr1, n);
}
```

Output:


```
The given array is :   1   2   3   4   5   6
The product array is: 720  360  240  180  144  120
```

67. Write a program in C to search an element in a row wise and column wise sorted matrix.

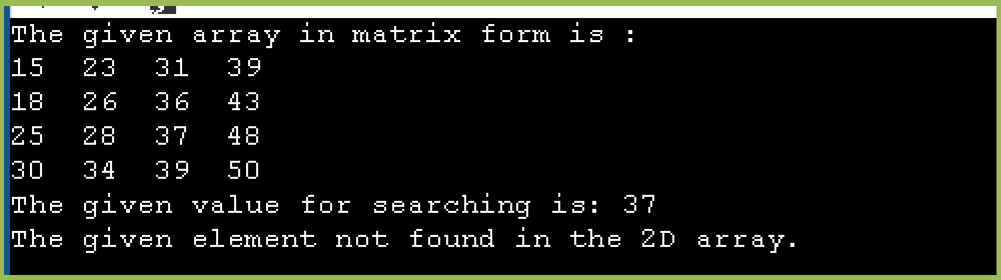
Ans

```
#include <stdio.h>

int searchElement(int arr2D[4][4], int n, int x)

{

  int i = 0, j = n-1;

  while ( i< n && j >= 0 )

  {

    if ( arr2D[i][j] == x )

    {

printf("\nThe element Found at the position in the matrix is: %d, %d", i, j);

      return 1;

    }

    if ( arr2D[i][j] < x )
```

```
        j--;
      else
        i++;
   }
printf("\nThe given element not found in the 2D array.");
   return 0;
}

int main()
{
  int arr2D[4][4] = { {15, 23, 31, 39},
                      {18, 26, 36, 43},
                      {25, 28, 37, 48},
                      {30, 34, 39, 50},
                    };
int i,j,v;
v=37;
```

```
        printf("The given array in matrix form is :  \n");
        for(i = 0; i < 4; i++)
        {
        for (j=0;j<4;j++)
        {
        printf("%d  ", arr2D[i][j]);
    }
        printf("\n");
        }

printf("The given value for searching is: %d",v);
searchElement(arr2D, 4, v);
  return 0;
}
```

Output:



```
The given array in matrix form is :
15  23  31  39
18  26  36  43
25  28  37  48
30  34  39  50
The given value for searching is: 37
The given element not found in the 2D array.
```

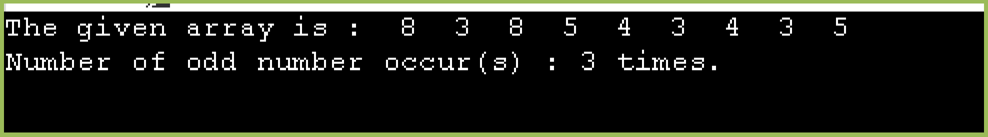
70. Write a program in C to find all numbers that occur odd number of times in an array.

Ans

```
#include <stdio.h>


int findOddCountElem (int *arr1, int n )

{

    int i, ResultXor = 0;

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

         {

      ResultXor = ResultXor ^ arr1[i];

          }

    return ResultXor;

}


int main()

{

   int i;

 int arr1[] = {8, 3, 8, 5, 4, 3, 4, 3, 5};
```

```
    int ctr = sizeof(arr1)/sizeof(arr1[0]);
printf("The given array is :  ");


        for(i = 0; i < ctr; i++)
        {
        printf("%d  ", arr1[i]);
    }
    printf("\n");


printf("Number of odd number occur(s) : %d times.\n",
findOddCountElem(arr1, ctr));
 return 0;
}
```

Output:



```
The given array is :  8  3  8  5  4  3  4  3  5
Number of odd number occur(s) : 3 times.
```

71. Write a program in C to find the median of two sorted arrays of different size.

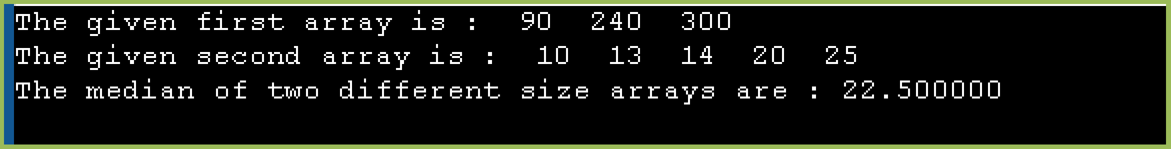
Ans

```
#include <stdio.h>
int findMax(int arr1, int arr2);
int findMin(int arr1, int arr2);

double medianOfDiffSortArrays(int *arr1, int n,
                              int *arr2, int m)
{

    int indexMin  = 0, indexMax  = n, i, j, median;
    while (indexMin <= indexMax)
    {
        i = (indexMin + indexMax) / 2;
        j = ((n + m + 1) / 2) - i;
        if (i < n && j > 0 && arr2[j - 1] > arr1[i])
            indexMin = i + 1;
        else if (i > 0 && j < m && arr2[j] < arr1[i - 1])
            indexMax = i - 1;
        else
        {
            if (i == 0)
                median = arr2[j - 1];
            else if (j == 0)
                median = arr1[i - 1];
            else
                median = findMax(arr1[i - 1], arr2[j - 1]);
            break;
        }
    }
    if ((n + m) % 2 == 1)
        return (double)median;
    if (i == n)
        return (median+arr2[j]) / 2.0;
    if (j == m)
        return (median + arr1[i]) / 2.0;
    return (median + findMin(arr1[i], arr2[j])) / 2.0;
```

```
}
int findMax(int arr1, int arr2)
{
    return arr1 > arr2 ? arr1 : arr2;
}
int findMin(int arr1, int arr2)
{
    return arr1 < arr2 ? arr1 : arr2;
}
int main()
{
    int arr1[] = {90, 240, 300};
    int arr2[] = { 10, 13, 14, 20, 25};
    int n = sizeof(arr1) / sizeof(int);
    int m = sizeof(arr2) / sizeof(int);
    int i;

        printf("The given first array is :  ");
        for(i = 0; i < n; i++)

        {

        printf("%d  ", arr1[i]);

    }

        printf("\n");

        printf("The given second array is :  ");

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

        {

        printf("%d  ", arr2[i]);

    }

        printf("\n");


    if (n < m)

printf("The median of two different size arrays are :
%f",medianOfDiffSortArrays(arr1, n, arr2, m));

    else

printf("The median of two different size arrays are :
%f",medianOfDiffSortArrays(arr2, m, arr1, n));

    return 0;
```

}

Output:



```
The given first array is :   90   240   300
The given second array is :   10   13   14   20   25
The median of two different size arrays are : 22.500000
```

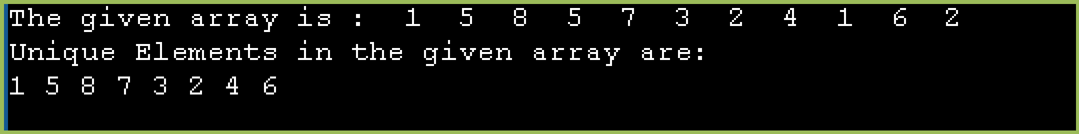
73. Write a program in C to print all unique elements of an unsorted array.

Ans

```
#include <stdio.h>

int main()

{

int arr1[]={1, 5, 8, 5, 7, 3, 2, 4, 1, 6, 2};

   int n = sizeof(arr1) / sizeof(int);

       int i, j;


       printf("The given array is :  ");

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

       {

       printf("%d  ", arr1[i]);
   }
       printf("\n");

       printf("Unique Elements in the given array are: \n");
       for(i = 0; i < n; i++)
       {
               for (j=0; j<i; j++)
               {
                       if (arr1[i] == arr1[j])
                       break;
     }

                       if (i == j)
                       {
                               printf("%d ", arr1[i]);
                       }
       }
return 0;
```

}

Output:



```
The given array is :   1   5   8   5   7   3   2   4   1   6   2
Unique Elements in the given array are:
1 5 8 7 3 2 4 6
```

74. Write a program in C to find the sum of upper triangular elements of a matrix.

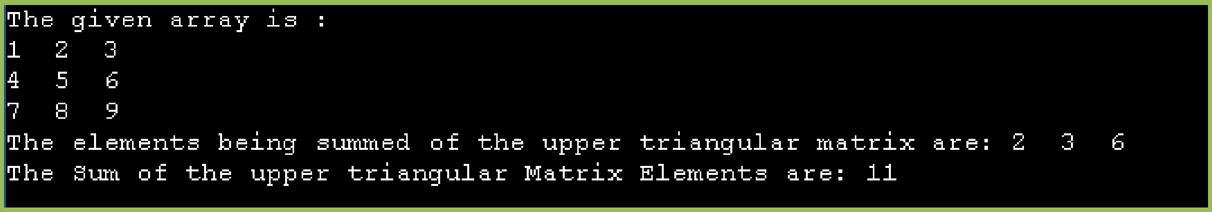
Ans

```
#include <stdio.h>

int main()

{

   int R, C, n, r, c, sum=0;

       int arr1[3][3]={{1, 2, 3},

                                    {4, 5, 6},

                                    {7, 8, 9}};

   R = C = n = 3;

       int i,j;


       printf("The given array is :  \n");
```

```
        for(i = 0; i < R; i++)

        {

        for (j=0;j<C;j++)

        {

        printf("%d  ", arr1[i][j]);

    }

        printf("\n");

        }


printf("The elements being summed of the upper triangular
matrix are: ");

for(r = 0; r < R; r++)

        {

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

                {

if(r < c)

                        {

                        printf("%d  ",arr1[r][c]);

               sum += arr1[r][c];
```

```
            }
        }
    }
printf("\nThe Sum of the upper triangular Matrix Elements are:
%d", sum);
    return 0;
}
```

Output:



```
The given array is :
1  2  3
4  5  6
7  8  9
The elements being summed of the upper triangular matrix are: 2  3  6
The Sum of the upper triangular Matrix Elements are: 11
```

75. Write a program in C to find the sum of lower triangular elements of a matrix.

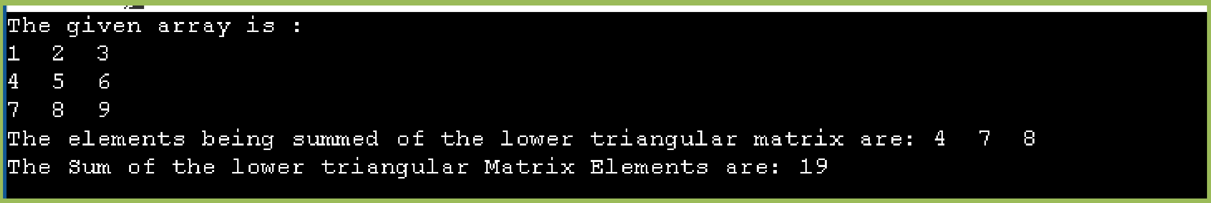
Ans

```
#include <stdio.h>
int main()
{
    int R, C, n, r, c, sum=0;
```

```
        int arr1[3][3]={{1, 2, 3},
                        {4, 5, 6},
                        {7, 8, 9}};
    R = C = n = 3;
        int i,j;

        printf("The given array is :  \n");
        for(i = 0; i < R; i++)
        {
        for (j=0;j<C;j++)
        {
        printf("%d  ", arr1[i][j]);
    }
        printf("\n");
        }

printf("The elements being summed of the lower triangular
matrix are: ");
for(r = 0; r < R; r++)
```

```
        {
for(c = 0; c < C; c++)
                {
if(r > c)
                        {
                        printf("%d  ",arr1[r][c]);
               sum += arr1[r][c];
            }
        }
    }
printf("\nThe Sum of the lower triangular Matrix Elements are:
%d", sum);
    return 0;
}
```

Output:



```
The given array is :
1  2  3
4  5  6
7  8  9
The elements being summed of the lower triangular matrix are: 4  7  8
The Sum of the lower triangular Matrix Elements are: 19
```

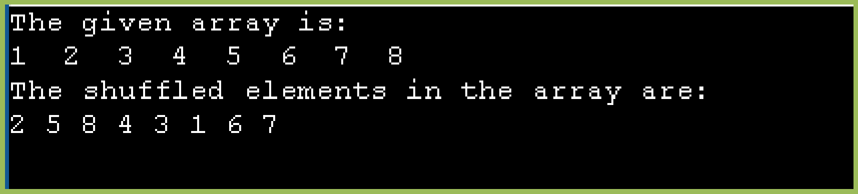
77. Write a program in C to generate a random permutation of array elements.

Ans

```
#include <stdio.h>

#include <stdlib.h>

#include <time.h>


void changeValues (int *a, int *b)

{

   int temp = *a;

   *a = *b;

   *b = temp;

}


void ArrayDisplay (int arr1[], int n)

{

        printf("The shuffled elements in the array are:  \n");

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

printf("%d ", arr1[i]);
```

```
    printf("\n");
}
void shuffleRandon ( int arr1[], int n )
{
    srand ( time(NULL) );
    for (int i = n-1; i > 0; i--)
    {
        int j = rand() % (i+1);
        changeValues(&arr1[i], &arr1[j]);
    }
}
int main()
{
    int arr1[] = {1, 2, 3, 4, 5, 6, 7, 8};
    int n = sizeof(arr1)/ sizeof(arr1[0]);
        int i;

        printf("The given array is:  \n");
        for(i = 0; i < n; i++)

                {

                        printf("%d  ", arr1[i]);

                }

        printf("\n");


    shuffleRandon (arr1, n);

ArrayDisplay(arr1, n);

    return 0;

}
```

Output:



```
The given array is:
1   2   3   4   5   6   7   8
The shuffled elements in the array are:
2 5 8 4 3 1 6 7
```

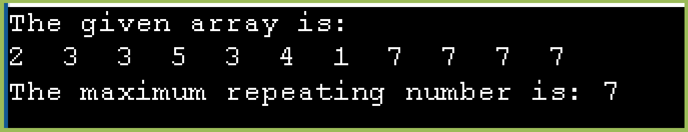
81. Write a program in C to find the maximum repeating number in a given array.

Ans

```
#include <stdio.h>

int numToRepeatMax(int* arr1 , int n, int k)
```

```
{
   int mx = arr1[0], result = 0;
       for (int i = 0; i< n; i++)
      arr1[arr1[i]%k] += k;

   for (int i = 1; i < n; i++)
   {
      if (arr1[i] > mx)
      {
         mx = arr1[i];
         result = i;
      }
   }
   return result;
}

int main()
{
   int arr1[] = {2, 3, 3, 5, 3, 4, 1, 7, 7, 7, 7};

   int n = sizeof(arr1)/sizeof(arr1[0]);

       int i;


       printf("The given array is:  \n");

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

              {

                     printf("%d  ", arr1[i]);

              }

       printf("\n");


   int k = 8;

printf("The maximum repeating number is: %d",
numToRepeatMax(arr1, n, k));

   return 0;

}
```

Output:



```
The given array is:
2   3   3   5   3   4   1   7   7   7   7
The maximum repeating number is: 7
```

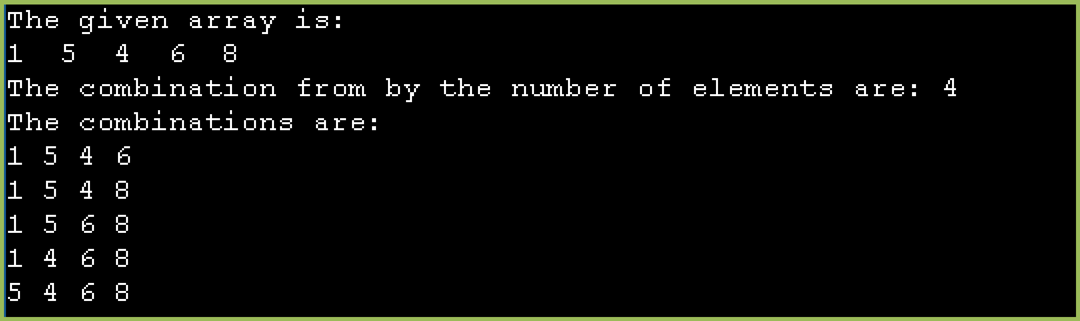
82. Write a program in C to print all possible combinations of r elements in a given array.

Ans

```
#include <stdio.h>

void makeCombination(int arr1[], int data[], int st, int end,

                    int index, int r);


void CombinationDisplay(int arr1[], int n, int r)

{

    int data[r];

makeCombination(arr1, data, 0, n-1, 0, r);

}


void makeCombination(int arr1[], int data[], int st, int end,

                    int index, int r)

{


    if (index == r)
    {
        for (int j=0; j<r; j++)
printf("%d ", data[j]);
        printf("\n");
        return;
    }
    for (int i=st; i<=end && end-i+1 >= r-index; i++)
    {
        data[index] = arr1[i];
makeCombination(arr1, data, i+1, end, index+1, r);
    }
}
int main()
{
    int arr1[] = {1, 5, 4, 6, 8};
    int r = 4,i;
    int n = sizeof(arr1)/sizeof(arr1[0]);
```

```
        printf("The given array is:  \n");
        for(i = 0; i < n; i++)
                {
                        printf("%d  ", arr1[i]);
                }
        printf("\n");


        printf("The combination from by the number of elements are: %d\n",r);
        printf("The combinations are: \n");
CombinationDisplay(arr1, n, r);
}
```

Output:



```
The given array is:
1   5   4   6   8
The combination from by the number of elements are: 4
The combinations are:
1 5 4 6
1 5 4 8
1 5 6 8
1 4 6 8
5 4 6 8
```

83. Write a program in C to find a pair with the given difference.

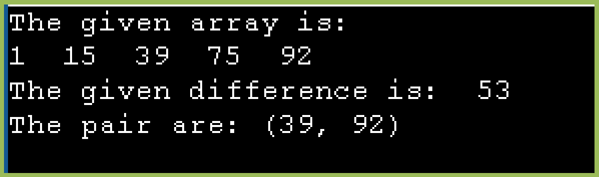
Ans

Source code:

```
#include <stdio.h>

# include <stdbool.h>


bool pairFinding(int arr1[], int size, int d)

{

    int i = 0;

    int j = 1;


    while (i<size && j<size)

    {

        if (i != j && arr1[j]-arr1[i] == d)

        {

printf("The pair are: (%d, %d)", arr1[i], arr1[j]);

            return true;

        }
        else if (arr1[j]-arr1[i] < d)
           j++;
        else
           i++;
    }

printf("No such pair found in the given array.");
    return false;
}

int main()
{
    int arr1[] = {1, 15, 39, 75, 92};
    int size = sizeof(arr1)/sizeof(arr1[0]);
    int d = 53,i;

         printf("The given array is:  \n");
         for(i = 0; i < size; i++)
```

```
                {
                        printf("%d  ", arr1[i]);
                }
        printf("\n");

        printf("The given difference is:  %d\n",d);
pairFinding(arr1, size, d);
    return 0;
}
```

Output:



```
The given array is:
1  15  39  75  92
The given difference is:  53
The pair are: (39, 92)
```

84. Write a program in C to find the minimum distance between two numbers in a given array.

Ans

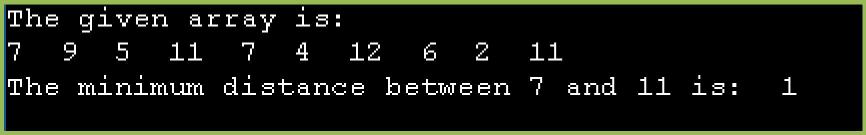
Source code:

```
#include <stdio.h>
#include <stdlib.h>
```

```
#include <limits.h>


int findMinDistance(int *input, int n1, int n2, int length)
{
    int pos_one = INT_MAX;
    int pos_two = INT_MAX;
    int d = length+1;
    int newD;
    pos_one = pos_two = d = length;


    for (int i = 0; i < length; i++)
    {
        if (input[i] == n1)
            pos_one = i;
        else if (input[i] == n2)
            pos_two = i;


        if (pos_one < length && pos_two < length)
        {
            newD = abs(pos_one - pos_two);
            if (d > newD)
                d = newD;
        }
    }

    return d == length+1 ? -1 : d;
}

int main()
{
    int arr1[] ={7, 9, 5, 11, 7, 4, 12, 6, 2, 11};
    int n = sizeof(arr1)/sizeof(arr1[0]);
    int p = 7;
    int q = 11,i;

        printf("The given array is:  \n");
        for(i = 0; i < n; i++)
            {
                printf("%d  ", arr1[i]);
            }
        printf("\n");
```

```
printf("The minimum distance between %d and %d is:  %d\n", p, q, findMinDistance(arr1, p, q, n));
  return 0;
}
```

Output:



```
The given array is:
7   9   5   11   7   4   12   6   2   11
The minimum distance between 7 and 11 is:   1
```

92. Write a program in C that checks whether the elements in an array are sorted or not.

Ans

Source code:

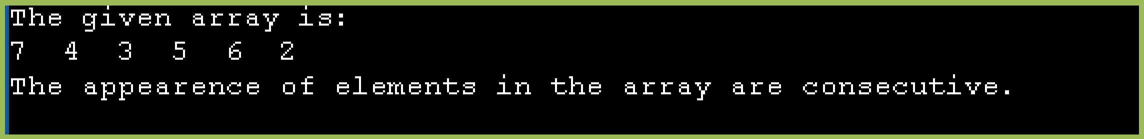
```
#include<stdio.h>

#include<stdlib.h>
```

```
#include <stdbool.h>


int FindMin(int arr1[], int n);

int FindMax(int arr1[], int n);


bool areConsecutive(int arr1[], int n)

{

  if ( n<  1 )

    return false;

  int min_no = FindMin(arr1, n);

  int max_no = FindMax(arr1, n);

  if (max_no - min_no  + 1 == n)

  {

      bool *checked = (bool *) calloc (n, sizeof(bool));

      int i;

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

      {

         if ( checked[arr1[i] - min_no] != false )
```

```
          return false;

        checked[arr1[i] - min_no] = true;

      }

      return true;

  }

  return false;

}


int FindMin(int arr1[], int n)

{

  int min_no = arr1[0];

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

   if (arr1[i] < min_no)

     min_no = arr1[i];

  return min_no;

}


int FindMax(int arr1[], int n)
```

```
{
  int max_no = arr1[0];
  for (int i = 1; i < n; i++)
   if (arr1[i] > max_no)
     max_no = arr1[i];
  return max_no;
}

int main()
{
    int arr1[]= {7, 4, 3, 5, 6, 2};

           int i;
           int arr_size = sizeof(arr1)/sizeof(arr1[0]);

         printf("The given array is:  \n");
         for(i = 0; i < arr_size; i++)
                  {
                        printf("%d  ", arr1[i]);
                }
        printf("\n");

    int n = sizeof(arr1)/sizeof(arr1[0]);
if(areConsecutive(arr1, n) == true)
printf("The appearence of elements in the array are
consecutive.");
    else
printf("The appearence of elements in the array are not
consecutive.");
    return 0;
}
```

Output:



```
The given array is:
7   4   3   5   6   2
The appearence of elements in the array are consecutive.
```

93. Write a program in C to rearrange positive and negative numbers alternatively in a given array.

Ans

Source code:

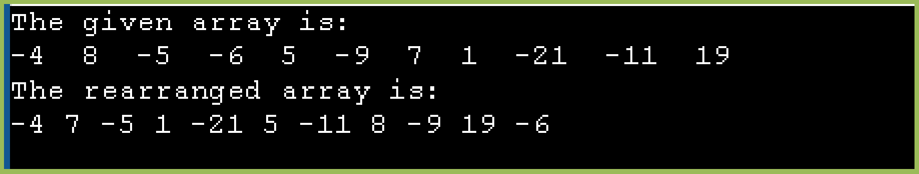
```
#include <stdio.h>

void changeNumber (int *arr1, int i, int j)
{
   int temp = arr1[i];
   arr1[i] = arr1[j];
   arr1[j] = temp;
}

void splitPosNeg(int *arr1, int size)
{
   int temp, left = 0, right = size-1;
while(right > left)
       {
    while(arr1[left] < 0)
```

```
         left++;
     while(arr1[right] > 0)
         right--;
if(left < right)
          {
changeNumber(arr1, left, right);
        }
    }
}

void rearrangeNumbers(int *arr1, int size)
{
    int i, j;
splitPosNeg(arr1, size);
for(i = 0; arr1[i] < 0; i++);
for(j = 1; (j < i) && (arr1[j] < 0); j += 2)
         {
changeNumber(arr1, i, j);
```

```
        i++;
    }
    return;
}

int main()
{
    int i, arr1[] = {-4, 8, -5, -6, 5, -9, 7, 1, -21, -11, 19};
          int arr_size = sizeof(arr1)/sizeof(arr1[0]);

        printf("The given array is:  \n");
        for(i = 0; i < arr_size; i++)
                {
                        printf("%d  ", arr1[i]);
                }
        printf("\n");

        printf("The rearranged array is:  \n");
```

```
rearrangeNumbers(arr1, 10);

for(i = 0; i < 11; i++){

printf("%d ", arr1[i]);

   }

   return 0;

}
```

Output:



```
The given array is:
-4   8   -5   -6   5   -9   7   1   -21   -11   19
The rearranged array is:
-4 7 -5 1 -21 5 -11 8 -9 19 -6
```

95. Write a program in C to segregate 0s and 1s in an array.

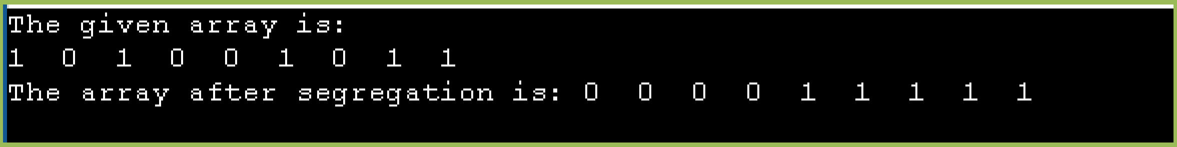
Ans

Source code:

```
#include <stdio.h>


void segZeroAndOne(int arr1[], int n)

{

   int ctr = 0;
```

```
    for (int i = 0; i < n; i++) {
        if (arr1[i] == 0)
            ctr++;
    }
    for (int i = 0; i < ctr; i++)
        arr1[i] = 0;

    for (int i = ctr; i < n; i++)
        arr1[i] = 1;
}
void printSegre(int arr1[], int n)
{
printf("The array after segregation is: ");
    for (int i = 0; i < n; i++)
printf("%d  ",arr1[i]);
}
int main()
{
    int arr1[] = { 1, 0, 1, 0, 0, 1, 0, 1, 1 };

    int n = sizeof(arr1) / sizeof(arr1[0]);

        int i;


        printf("The given array is:  \n");

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

                {

                        printf("%d  ", arr1[i]);

                }

        printf("\n");


segZeroAndOne(arr1, n);

printSegre(arr1, n);


    return 0;

}
```

Output:


```
The given array is:
1  0  1  0  0  1  0  1  1
The array after segregation is: 0  0  0  0  1  1  1  1  1
```

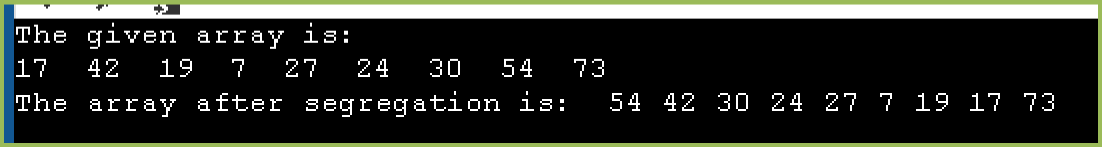
96. Write a program in C to segregate even and odd elements on an array.

Ans

Source code:

```
#include<stdio.h>


void changePlace (int *ar, int *br);

void EvenOddSegre(int arr1[], int size)

{

    int l_index = 0, r_index = size-1;

    while (l_index < r_index)

    {

        while (arr1[l_index]%2 == 0 && l_index < r_index)

            l_index++;

        while (arr1[r_index]%2 == 1 && l_index < r_index)

            r_index--;

        if (l_index < r_index)
        {
            changePlace(&arr1[l_index], &arr1[r_index]);
            l_index++;
            r_index--;
        }
    }
}

void changePlace(int *ar, int *br)
{
    int temp = *ar;
    *ar = *br;
    *br = temp;
}

int main()
{
    int arr1[] = {17, 42, 19, 7, 27, 24, 30, 54, 73};
    int arr_size = sizeof(arr1)/sizeof(arr1[0]);

    int i = 0;


        printf("The given array is:  \n");

        for(i = 0; i < arr_size; i++)

                {

                        printf("%d  ", arr1[i]);

                }

        printf("\n");


EvenOddSegre(arr1, arr_size);

printf("The array after segregation is:  ");

    for (i = 0; i < arr_size; i++)

printf("%d ", arr1[i]);

    return 0;

}
```

Output:


```
The given array is:
17   42   19   7   27   24   30   54   73
The array after segregation is:   54 42 30 24 27 7 19 17 73
```

102.Write a program in C to rearrange an array in such an order that– smallest, largest, 2nd smallest, 2nd largest and on.

Ans

Source code:

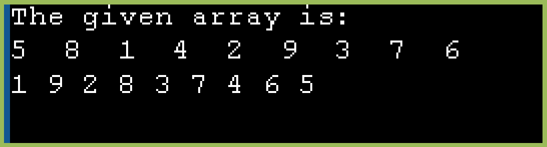
```
#include<stdio.h>

void sort(int arr1[], int n)

{

  int i, j,temp;

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

      {

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

    {

       if (arr1[j] > arr1[j+1])

       {

          temp = arr1[j];

          arr1[j] = arr1[j+1];
```

```
            arr1[j+1] = temp;
          }
        }
         }

}

void rearrangeArray(int arr1[], int n)
{
sort(arr1, n);
   int tempArr[n];
   int ArrIndex = 0;
   for (int i = 0, j = n-1; i <= n / 2 || j > n / 2; i++, j--)
       {
      tempArr[ArrIndex] = arr1[i];
      ArrIndex++;
      tempArr[ArrIndex] = arr1[j];
      ArrIndex++;
```

```
    }
    for (int i = 0; i < n; i++)
        {arr1[i] = tempArr[i];}
}

int main()
{
    int arr1[] = { 5, 8, 1, 4, 2, 9, 3, 7, 6 };
    int n = sizeof(arr1) / sizeof(arr1[0]);
            int i = 0;

        printf("The given array is:  \n");
        for(i = 0; i < n; i++)
                {
                        printf("%d  ", arr1[i]);
                }
        printf("\n");
```

```
rearrangeArray(arr1, n);
   for (int i = 0; i < n; i++)
printf("%d ",arr1[i]);
   return 0;
}
```

Output:



```
The given array is:
5  8  1  4  2  9  3  7  6
1 9 2 8 3 7 4 6 5
```

103.Write a program in C to update every array element with multiplication of previous and next numbers in array.

Ans

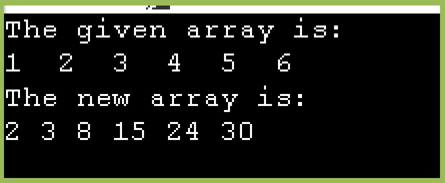
Source code:

```
#include<stdio.h>
void newArryPrevNext(int arr1[], int n)
{
   if (n <= 1)
     return;
```

```
    int pre_elem = arr1[0];

    arr1[0] = arr1[0] * arr1[1];

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

    {

        int cur_elem = arr1[i];

        arr1[i] = pre_elem * arr1[i+1];

        pre_elem = cur_elem;

    }

    arr1[n-1] = pre_elem * arr1[n-1];

}

int main()

{

    int arr1[] = {1,2, 3, 4, 5, 6};

    int n = sizeof(arr1)/sizeof(arr1[0]);

        int i = 0;


        printf("The given array is:  \n");

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

```
                {
                        printf("%d  ", arr1[i]);
                }
        printf("\n");

        printf("The new array is: \n");
newArryPrevNext(arr1, n);
    for (int i=0; i<n; i++)
printf("%d ", arr1[i]);
    return 0;
}
```

Output:



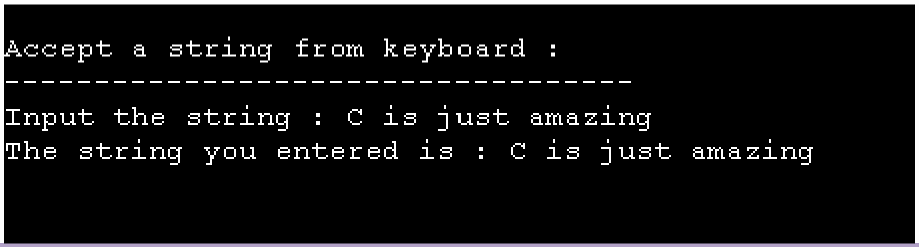
```
The given array is:
1   2   3   4   5   6
The new array is:
2 3 8 15 24 30
```

Functions

1. Write a program in C to show the simple structure of a function.

```
#include <stdio.h>

#include <stdlib.h>


void main()

{

    char str[50];


printf("\n\nAccept a string from keyboard
:\n");

printf("-----------------------------------
-\n");

printf("Input the string : ");

fgets(str, sizeof str, stdin);

printf("The string you entered is : %s\n",
str);
}
```

Output:



```
Accept a string from keyboard :
-----------------------------------
Input the string : C is just amazing
The string you entered is : C is just amazing
```

2. Write a program in C to find the square of any number using the function.

Test Data:

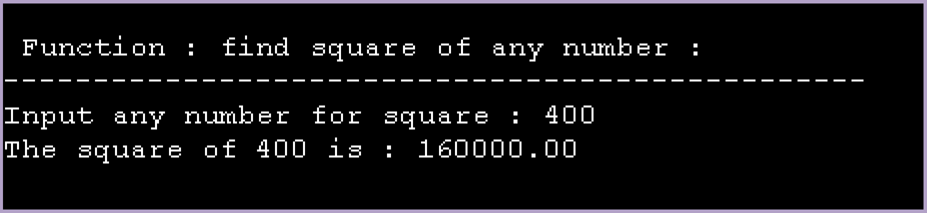
Input any number for square: 20

Expected Output :

The square of 20 is : 400.00

```
#include <stdio.h>


double square(double num)

{
    return (num * num);
}
int main()
{
    int num;
    double n;
        printf("\n\n Function : find square of any number
:\n");
        printf("------------------------------------------------\n");

    printf("Input any number for square : ");
    scanf("%d", &num);
    n = square(num);
    printf("The square of %d is : %.2f\n", num, n);
    return 0;
}
```

Output:



```
 Function : find square of any number :
------------------------------------------------
Input any number for square : 400
The square of 400 is : 160000.00
```

3. Write a program in C to swap two numbers using function.

Test Data :

Input 1st number : 2

Input 2nd number : 4

Expected Output :

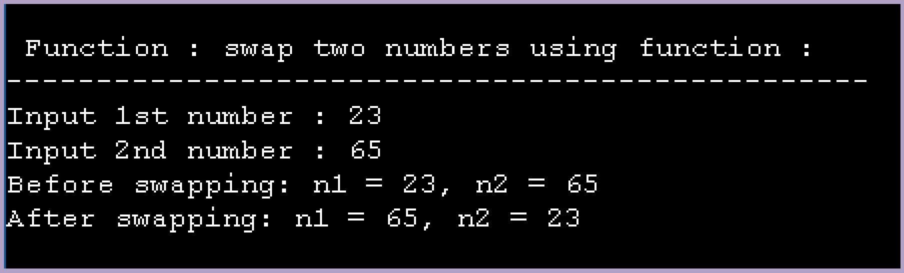
Before swapping: n1 = 2, n2 = 4

After swapping: n1 = 4, n2 = 2

```
#include<stdio.h>

void swap(int *,int *);
int main()
{

    int n1,n2;
   printf("\n\n Function : swap two 
numbers using function :\n");
```

```
   printf("---------------------------------
-----------------\n");
printf("Input 1st number : ");
scanf("%d",&n1);
printf("Input 2nd number : ");
scanf("%d",&n2);

printf("Before swapping: n1 = %d, n2 = %d
",n1,n2);
   //pass the address of both variables to
the function.
    swap(&n1,&n2);

printf("\nAfter swapping: n1 = %d, n2 = %d
\n\n",n1,n2);
    return 0;
}
```

Output:



```
 Function : swap two numbers using function :
------------------------------------------------
Input 1st number : 23
Input 2nd number : 65
Before swapping: n1 = 23, n2 = 65
After swapping: n1 = 65, n2 = 23
```

4. Write a program in C to check a given number is even or odd using the function.

Test Data :

Input any number : 5

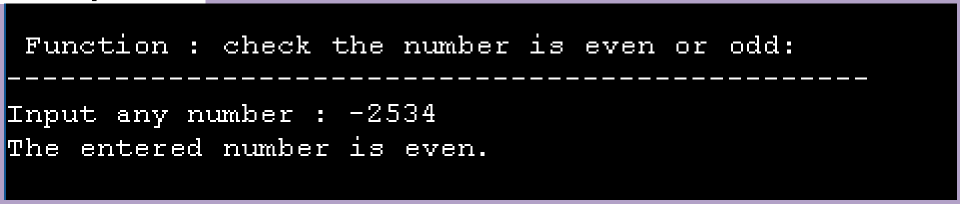
Expected Output :

The entered number is odd.

```
#include <stdio.h>

//if the least significant bit is 1 the
number is odd and 0 the number is even
int checkOddEven(int n1)
{
    return (n1 & 1);//The & operator does
a bitwise and,
}

int main()
{
    int n1;
   printf("\n\n Function : check the
number is even or odd:\n");
   printf("-------------------------------
-----------------\n");
printf("Input any number : ");
scanf("%d", &n1);
```

```
    // If checkOddEven() function returns
1 then the number is odd
    if(checkOddEven(n1))
    {
printf("The entered number is odd.\n\n");
    }
    else
    {
printf("The entered number is even.\n\n");
    }
    return 0;
}
```

Output:



```
 Function : check the number is even or odd:
------------------------------------------------
Input any number : -2534
The entered number is even.
```

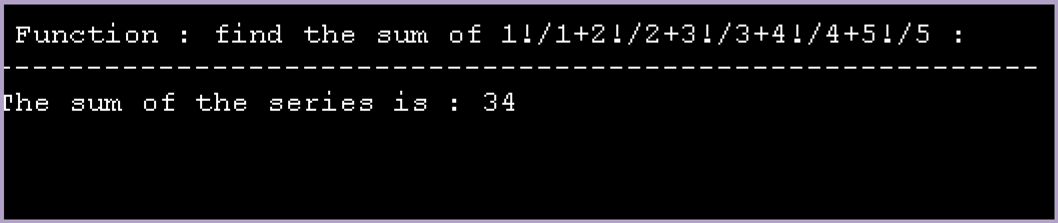
5. Write a program in C to find the sum of the series 1!/1+2!/2+3!/3+4!/4+5!/5 using the function.

Expected Output :

The sum of the series is : 34

```
#include <stdio.h>
```

```
int fact(int);
void main()
{
    int sum;

sum=fact(1)/1+fact(2)/2+fact(3)/3+fact(4)/
4+fact(5)/5;
   printf("\n\n Function : find the sum of
1!/1+2!/2+3!/3+4!/4+5!/5 :\n");
   printf("-------------------------------
-------------------------------\n");
printf("The sum of the series is :
%d\n\n",sum);
}

int fact(int n)
    {
        int num=0,f=1;
        while(num<=n-1)
        {
            f =f+f*num;
num++;
        }
    return f;
    }
Output:
```



```
Function : find the sum of 1!/1+2!/2+3!/3+4!/4+5!/5 :
----------------------------------------------------------
The sum of the series is : 34
```

6. Write a program in C to convert decimal number to binary number using the function.

Test Data :

Input any decimal number : 65

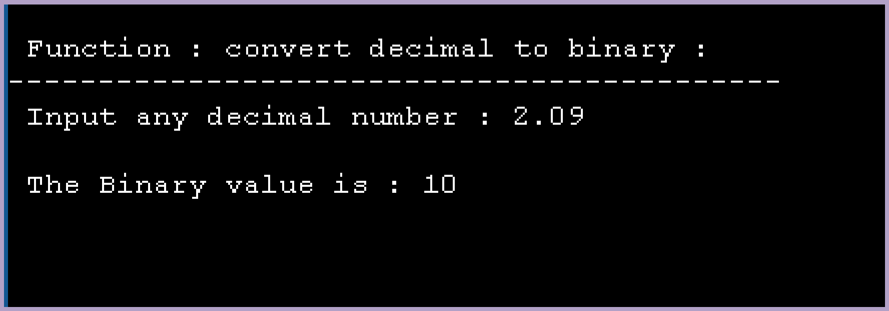
Expected Output :

The Binary value is : 1000001

```
#include<stdio.h>

longtoBin(int);

intmain()
{
longbno;
intdno;
    printf("\n\n Function : convert decimal
to binary :\n");
```

```
    printf("-----------------------------------
------------\n");
printf(" Input any decimal number : ");
scanf("%d",&dno);
bno=toBin(dno);
printf("\n The Binary value is :
%ld\n\n",bno);

return0;
}
longtoBin(intdno)
{
longbno=0,remainder,f=1;
while(dno!=0)
{
         remainder =dno%2;
bno=bno+ remainder * f;
         f = f *10;
dno=dno/2;
}
returnbno;
}
```

Output:



```
Function : convert decimal to binary :
------------------------------------------
Input any decimal number : 2.09

The Binary value is : 10
```

7. Write a program in C to check whether a number is a prime number or not using the function.

Test Data :

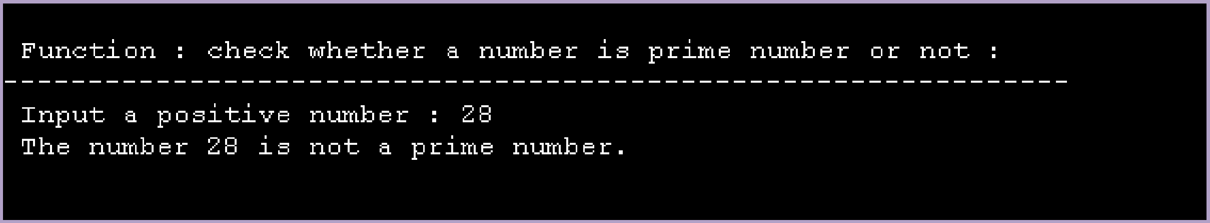
Input a positive number : 5

Expected Output :

The number 5 is a prime number.

```
#include<stdio.h>
intPrimeOrNot(int);
intmain()
{
int  n1,prime;
     printf("\n\n Function : check whether a
number is prime number or not :\n");
     printf("-------------------------------
------------------------------\n");
printf(" Input a positive number : ");
scanf("%d",&n1);
    prime =PrimeOrNot(n1);
if(prime==1)
printf(" The number %d is a prime
number.\n",n1);
else
printf(" The number %d is not a prime
number.\n",n1);
return0;
}
intPrimeOrNot(int n1)
{
inti=2;
while(i<=n1/2)
{
if(n1%i==0)
return0;
else
i++;
}
return1;
}
```

Output:



```
Function : check whether a number is prime number or not :
---------------------------------------------------------------
Input a positive number : 28
The number 28 is not a prime number.
```

8. Write a program in C to get the largest element of an array using the function.

Test Data :

Input the number of elements to be stored in the array :5

Input 5 elements in the array :

element - 0 : 1

element - 1 : 2

element - 2 : 3

element - 3 : 4

element - 4 : 5

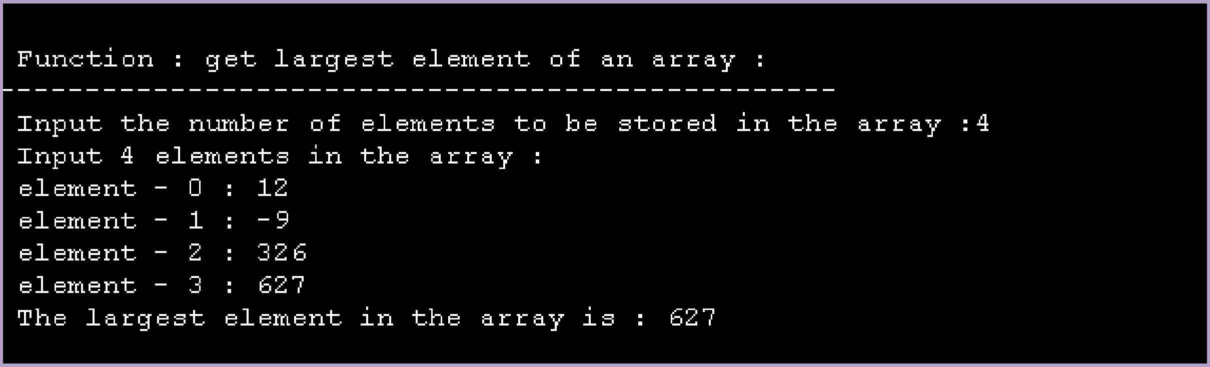
Expected Output :

The largest element in the array is : 5

```
#include<stdio.h>
#define MAX 100
```

```
intfindMaxElem(int[]);
int n;

intmain()
{
int arr1[MAX],mxelem,i;
    printf("\n\n Function : get largest
element of an array :\n");
    printf("---------------------------------
-----------------\n");

printf(" Input the number of elements to
be stored in the array :");
scanf("%d",&n);

printf(" Input %d elements in the array
:\n",n);
for(i=0;i<n;i++)
{
    printf(" element - %d : ",i);
    scanf("%d",&arr1[i]);
    }
mxelem=findMaxElem(arr1);

printf(" The largest element in the array
is : %d\n\n",mxelem);
return0;
}
intfindMaxElem(int arr1[])
{
inti=1,mxelem;
mxelem=arr1[0];
while(i< n)
    {
if(mxelem<arr1[i])
mxelem=arr1[i];
i++;
}
returnmxelem;
}
```

Output:



```
Function : get largest element of an array :
-----------------------------------------------
Input the number of elements to be stored in the array :4
Input 4 elements in the array :
element - 0 : 12
element - 1 : -9
element - 2 : 326
element - 3 : 627
The largest element in the array is : 627
```

9. Write a program in C to check armstrong and perfect numbers using the function.

Test Data :

Input any number: 371

Expected Output :

The 371 is an Armstrong number.

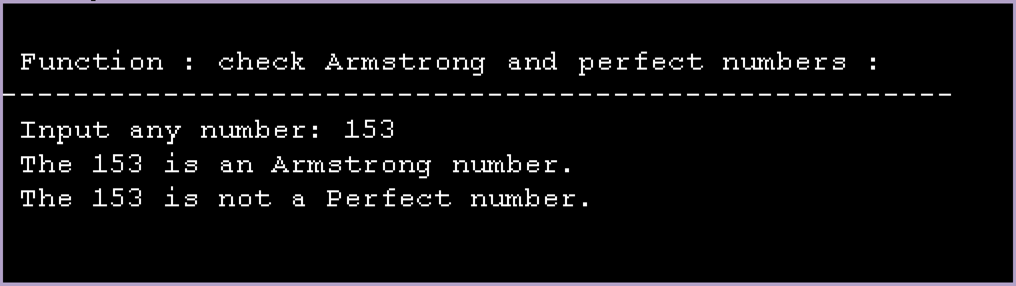
The 371 is not a Perfect number.

```
#include <stdio.h>

intcheckArmstrong(int n1);
intcheckPerfect(int n1);

intmain()
{
int n1;
    printf("\n\n Function : check Armstrong
and perfect numbers :\n");
    printf("-------------------------------------
----------------------\n");

printf(" Input any number: ");
scanf("%d",&n1);


//Calls the isArmstrong() function
if(checkArmstrong(n1))
{
printf(" The %d is an Armstrong
number.\n", n1);
}
else
{
printf(" The %d is not an Armstrong
number.\n", n1);
}

//Calls the checkPerfect() function
if(checkPerfect(n1))
{
printf(" The %d is a Perfect number.\n\n",
n1);
}
else
{
printf(" The %d is not a Perfect
number.\n\n", n1);
}
return0;
}

// Checks whether a three digits number is
Armstrong number or not.
```

```
//An Armstrong number is an n-digit number
that is equal
//to the sum of the n-th powers of its
digits.
intcheckArmstrong(int n1)
{
intld, sum,num;
    sum =0;
num= n1;
while(num!=0)
{
ld=num%10;// find the last digit of the
number
        sum +=ld*ld*ld;//calculate the
cube of the last digit and adds to sum
num=num/10;
}
return(n1 == sum);
}
// Checks whether the number is perfect
number or not.
//a perfect number is a positive integer
that is equal to
//the sum of its positive divisors
excluding the number itself
intcheckPerfect(int n1)
{
```

```
inti, sum,num;
    sum =0;
num= n1;
for(i=1;i<num;i++)
{
/* If i is a divisor of n1 */
if(num%i==0)
{
            sum +=i;
}
}
return(n1 == sum);
}
```

Output:



```
Function : check Armstrong and perfect numbers :
-----------------------------------------------------
Input any number: 153
The 153 is an Armstrong number.
The 153 is not a Perfect number.
```

10. Write a program in C to print all perfect numbers in given range using the function.

Test Data :

Input lowest search limit of perfect numbers : 1

Input lowest search limit of perfect numbers : 100

Expected Output :

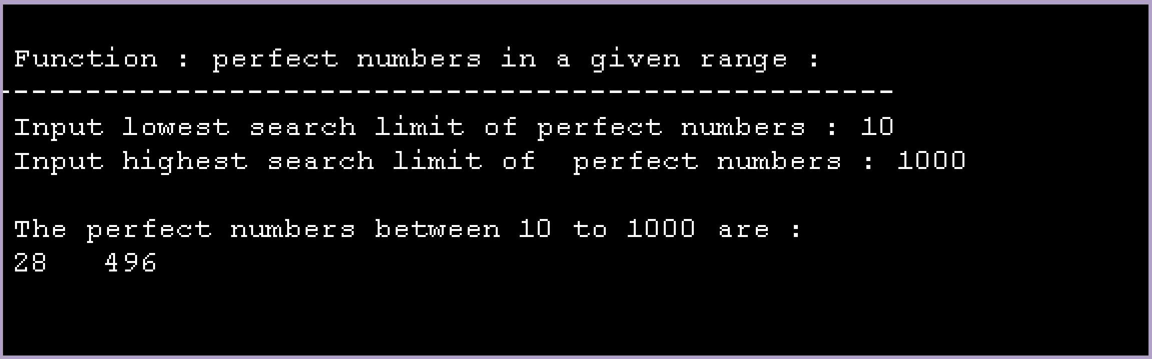
The perfect numbers between 1 to 100 are :

```
 6  28
```

```
#include <stdio.h>
/* Function declarations */
intcheckPerfect(int n1);
voidPerfectNumbers(intstLimit,intenLimit);

intmain()
{
intstLimit,enLimit;
    printf("\n\n Function : perfect numbers
in a given range :\n");
    printf("-----------------------------------
--------------------\n");
printf(" Input lowest search limit of
perfect numbers : ");
scanf("%d",&stLimit);
printf(" Input highest search limit of
perfect numbers : ");
scanf("%d",&enLimit);

printf("\n The perfect numbers between %d
to %d are : \n",stLimit,enLimit);
PerfectNumbers(stLimit,enLimit);
printf("\n\n");
return0;
}
// Checks whether the given number is
perfect or not.

intcheckPerfect(int n1)
{
inti, sum;

    sum =0;
for(i=1;i<n1;i++)
{
if(n1 %i==0)
{
            sum +=i;
}
}
// If sum of proper positive divisors
equals to given number
// then the number is perfect number
if(sum == n1)
return1;
else
return0;
}
```

```
voidPerfectNumbers(intstLimit,intenLimit)
{
/* print perfect numbers from start to end
*/
while(stLimit<=enLimit)
{
if(checkPerfect(stLimit))
{
printf(" %d  ",stLimit);
}
stLimit++;
}
}
```

Output:



```
Function : perfect numbers in a given range :
-------------------------------------------------
Input lowest search limit of perfect numbers : 10
Input highest search limit of  perfect numbers : 1000

The perfect numbers between 10 to 1000 are :
28   496
```

11. Write a program in C to check whether two given strings are an anagram.

Test Data :

Input the first String : spare

Input the second String : pears

Expected Output :

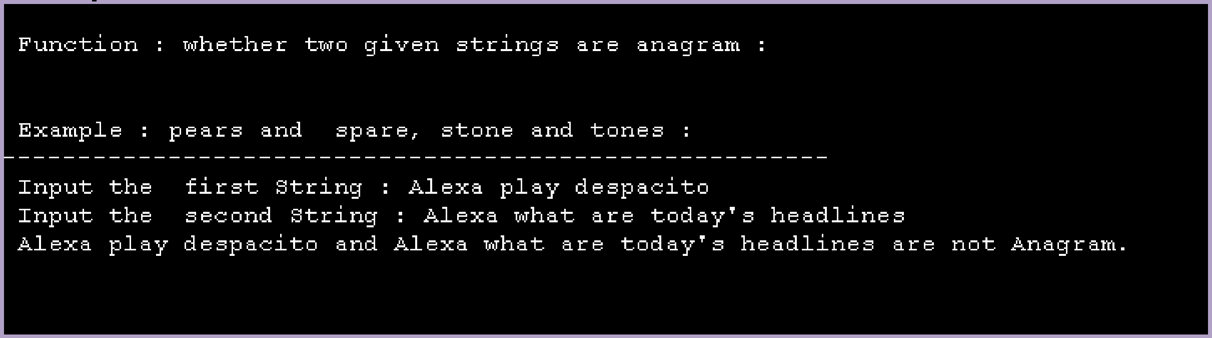
spare and pears are Anagram.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

//Two strings are anagram of each other,
if we can rearrange
//characters of one string to form another
string. All the characters
//of one string must be present in another
string and should appear same
//number of time in other string. Strings
can contain any ASCII characters.
//Example : rescued and secured, resign
and singer, stone and tones,
//pears and spare, ELEVEN PLUS TWO and
TWELVE PLUS ONE
```

```
intcheckAnagram(char*str1,char*str2);
intmain()
{
char str1[100], str2[100];
printf("\n\n Function : whether two given
strings are anagram :\n");
printf("\n\n Example : pears and  spare,
stone and tones :\n");
   printf("--------------------------------
------------------------\n");
printf(" Input the  first String : ");
fgets(str1,sizeof str1,stdin);
printf(" Input the  second String : ");
fgets(str2,sizeof str2,stdin);

if(checkAnagram(str1, str2)==1)
{
       str1[strlen(str1)-1]='\0';
       str2[strlen(str2)-1]='\0';
printf(" %s and %s are
Anagram.\n\n",str1,str2);
}
else
{
       str1[strlen(str1)-1]='\0';
       str2[strlen(str2)-1]='\0';
printf(" %s and %s are not
Anagram.\n\n",str1,str2);
}
return0;
}

//Function to check whether two passed
strings are anagram or not
intcheckAnagram(char*str1,char*str2)
{
int str1ChrCtr[256]={0},
str2ChrCtr[256]={0};
intctr;
/* check the length of equality of Two
Strings */
if(strlen(str1)!=strlen(str2))
{
return0;
}
//count frequency of characters in str1
for(ctr=0; str1[ctr]!='\0';ctr++)
{
        str1ChrCtr[str1[ctr]]++;
}
//count frequency of characters in str2
for(ctr=0; str2[ctr]!='\0';ctr++)
```

```
{
        str2ChrCtr[str2[ctr]]++;
}
//compare character counts of both strings
for(ctr=0;ctr<256;ctr++)
{
if(str1ChrCtr[ctr]!= str2ChrCtr[ctr])
return0;
}
return1;
}
```

Output:



```
Function : whether two given strings are anagram :


Example : pears and  spare, stone and tones :
-------------------------------------------------------
Input the  first String : Alexa play despacito
Input the  second String : Alexa what are today's headlines
Alexa play despacito and Alexa what are today's headlines are not Anagram.
```

12. Write a C programming to find out maximum and minimum of some values using function which will return an array.

Test Data :

Input 5 values

25

11

35

65

20

Expected Output :

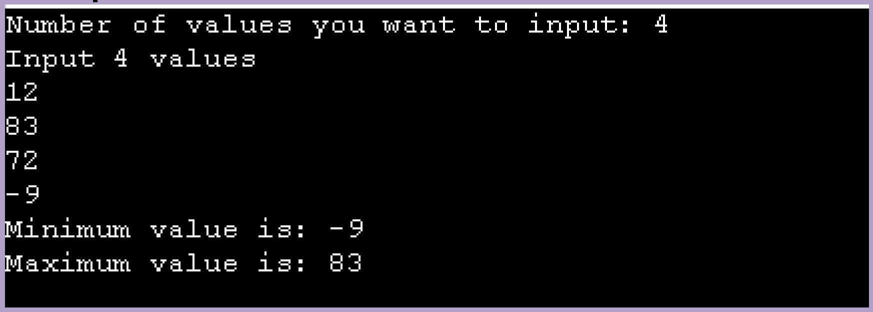
Number of values you want to input: Input 5 values

Minimum value is: 11

Maximum value is: 65

```
# include <stdio.h>
# define max 10
int*maxmin(intar[],int v);
intmain()
{
    intarr[max];
    intn,i,*p;
    printf("Number of values you want to
input: ");
    scanf("%d",&n);
    printf("Input %d values\n", n);
    for(i=0;i<n;i++)
        scanf("%d",&arr[i]);
    p=maxmin(arr,n);
    printf("Minimum value is: %d\n",*p++);
    printf("Maximum value is: %d\n",*p);
}
int*maxmin(int arra1[],int v)
{
    inti;
    staticintresult_mm[2];
    result_mm[0]=arra1[0];
    result_mm[1]=arra1[0];
    for(i=1;i<v;i++)
    {
        if(result_mm[0]> arra1[i])
            result_mm[0]=arra1[i];
        if(result_mm[1]< arra1[i])
            result_mm[1]= arra1[i];
    }
    returnresult_mm;
}
```

Output:

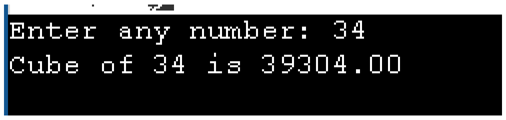


```
Number of values you want to input: 4
Input 4 values
12
83
72
-9
Minimum value is: -9
Maximum value is: 83
```

1.Write a C program to find cube of any number using function.

```
#include <stdio.h>

/* Function declaration */
double cube(double num);

int main()
{
   int num;
   double c;

   /* Input number to find cube from user */
printf("Enter any number: ");
scanf("%d", &num);

   c = cube(num);

printf("Cube of %d is %.2f", num, c);

   return 0;
}


/**
 * Function to find cube of any number
 */
double cube(double num)
{
   return (num * num * num);
}
```

Output:



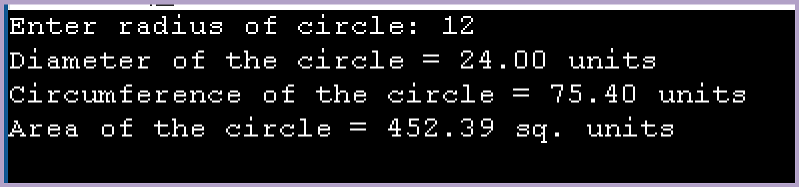
```
Enter any number: 34
Cube of 34 is 39304.00
```

2.Write a C program to find diameter, circumference and area of circle using functionsusing functions

```
#include <stdio.h>

#include <math.h> // Used for constant PI referred as M_PI




/* Function declaration */

double getDiameter(double radius);
double getCircumference(double radius);
double getArea(double radius);



int main()
{
    float radius, dia, circ, area;

    /* Input radius of circle from user */
printf("Enter radius of circle: ");
scanf("%f", &radius);

dia  = getDiameter(radius);       // Call getDiameter
function
circ = getCircumference(radius);  // Call
getCircumference function
    area = getArea(radius);           // Call getArea function
printf("Diameter of the circle = %.2f units\n", dia);

printf("Circumference of the circle = %.2f units\n", circ);

printf("Area of the circle = %.2f sq. units", area);


    return 0;

}




/**

 * Calculate diameter of circle whose radius is given

 */

double getDiameter(double radius)

{

    return (2 * radius);

}
```

```
/**
 * Calculate circumference of circle whose radius is given
 */
double getCircumference(double radius)
{
    return (2 * M_PI * radius); // M_PI = PI = 3.14 ...
}
double getArea(double radius)
{
    return (M_PI * radius * radius); // M_PI = PI = 3.14 ...
}
```

Output:



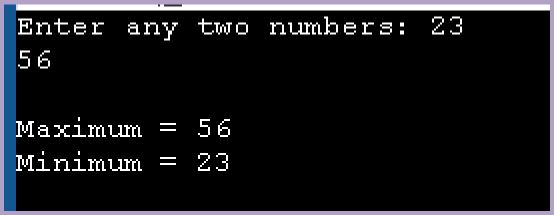
```
Enter radius of circle: 12
Diameter of the circle = 24.00 units
Circumference of the circle = 75.40 units
Area of the circle = 452.39 sq. units
```

3.Write a C program to find maximum and minimum between two numbers using functions.

```
#include <stdio.h>


/* Function declarations */
int max(int num1, int num2);
int min(int num1, int num2);
int main()
{
    int num1, num2, maximum, minimum;


    /* Input two numbers from user */
printf("Enter any two numbers: ");
scanf("%d%d", &num1, &num2);


    maximum = max(num1, num2);  // Call maximum
function
    minimum = min(num1, num2);  // Call minimum
function
printf("\nMaximum = %d\n", maximum);
printf("Minimum = %d", minimum);


    return 0;
}




/**
 * Find maximum between two numbers.
 */
int max(int num1, int num2)
{
    return (num1 > num2 ) ? num1 : num2;
}

/**
```

```
 * Find minimum between two numbers.
 */
int min(int num1, int num2)
{
    return (num1 > num2 ) ? num2 : num1;
}
```

Output:



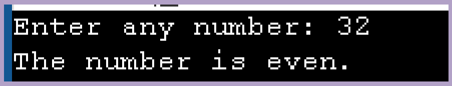
```
Enter any two numbers: 23
56

Maximum = 56
Minimum = 23
```

4.Write a C program to check whether a number is even or odd using functions.

```
#include <stdio.h>




/**
 * Function to check even or odd
 * Returns 1 is num is even otherwise 0
 */
int isEven(int num)
{
    return !(num& 1);
}



int main()
{
    int num;

    /* Input number from user */
printf("Enter any number: ");
scanf("%d", &num);
```

```
    /* If isEven() function returns 0 then the number is
even */
    if(isEven(num))
    {
printf("The number is even.");
    }
    else
    {
printf("The number is odd.");
    }


    return 0;
}
```

Output:



```
Enter any number: 32
The number is even.
```

5.Write a C program to check whether a number is prime, Armstrong or perfect number using functions.

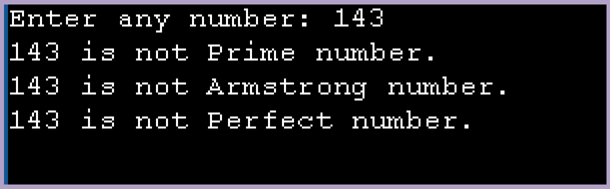
```
#include <stdio.h>

#include <math.h>



/* Function declarations */

int isPrime(int num);

int isArmstrong(int num);

int isPerfect(int num);



int main()

{

    int num;


printf("Enter any number: ");

scanf("%d", &num);
```

```
    // Call isPrime() functions
    if(isPrime(num))
    {
printf("%d is Prime number.\n", num);
    }
    else
    {
printf("%d is not Prime number.\n", num);
    }

    // Call isArmstrong() function
    if(isArmstrong(num))
    {
printf("%d is Armstrong number.\n", num);
    }
    else
    {
```

```
printf("%d is not Armstrong number.\n", num);
    }


    // Call isPerfect() function
    if(isPerfect(num))
    {
printf("%d is Perfect number.\n", num);
    }
    else
    {
printf("%d is not Perfect number.\n", num);
    }


    return 0;
}
```

```
/**
 * Check whether a number is prime or not.
 * Returns 1 if the number is prime otherwise 0.
 */
int isPrime(int num)
{
    int i;

    for(i=2; i<=num/2; i++)
    {
        /*
         * If the number is divisible by any number
         * other than 1 and self then it is not prime
         */
        if(num%i == 0)
        {
            return 0;
        }
    }

    return 1;
}




/**
 * Check whether a number is Armstrong number or not.
 * Returns 1 if the number is Armstrong number
otherwise 0.
 */
int isArmstrong(int num)
{
    int lastDigit, sum, originalNum, digits;
    sum = 0;

originalNum = num;

    /* Find total digits in num */
    digits = (int) log10(num) + 1;

    /*
     * Calculate sum of power of digits
     */
    while(num> 0)
    {
        // Extract the last digit
lastDigit = num % 10;

        // Compute sum of power of last digit
        sum = sum + round(pow(lastDigit, digits));
        // Remove the last digit
num = num / 10;
    }

    return (originalNum == sum);
}




/**
 * Check whether the number is perfect number or not.
 * Returns 1 if the number is perfect otherwise 0.
 */
int isPerfect(int num)
{
    int i, sum, n;
    sum = 0;
    n = num;


    for(i=1; i<n; i++)
    {
        /* If i is a divisor of num */
        if(n%i == 0)
        {
            sum += i;
        }
    }


    return (num == sum);
}
```

Output:



```
Enter any number: 143
143 is not Prime number.
143 is not Armstrong number.
143 is not Perfect number.
```

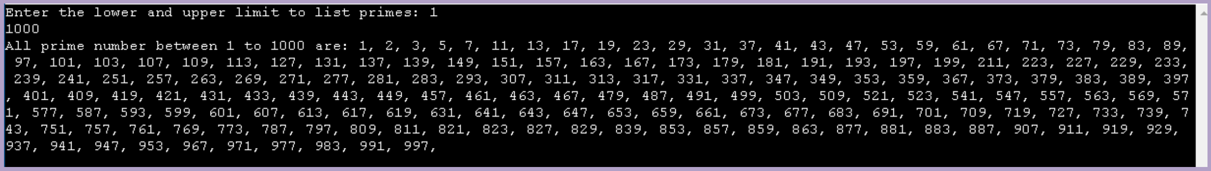
6.Write a C program to find all prime numbers between given interval using functions.

```
#include <stdio.h>



/* Function declarations */

int isPrime(int num);

void printPrimes(int lowerLimit, int upperLimit);




int main()

{

    int lowerLimit, upperLimit;


printf("Enter the lower and upper limit to list primes: ");

scanf("%d%d", &lowerLimit, &upperLimit);
```

```
    // Call function to print all primes between the given
range.
printPrimes(lowerLimit, upperLimit);


    return 0;
}





/**
 * Print all prime numbers between lower limit and upper
limit.
 */
void printPrimes(int lowerLimit, int upperLimit)
{
printf("All prime number between %d to %d are: ",
lowerLimit, upperLimit);
```

```
    while(lowerLimit<= upperLimit)
    {
        // Print if current number is prime.
        if(isPrime(lowerLimit))
        {
printf("%d, ", lowerLimit);
        }

lowerLimit++;
    }
}




/**
 * Check whether a number is prime or not.
 * Returns 1 if the number is prime otherwise 0.
 */
int isPrime(int num)
{
    int i;

    for(i=2; i<=num/2; i++)
    {
        /*
         * If the number is divisible by any number
         * other than 1 and self then it is not prime
         */
        if(num % i == 0)
        {
            return 0;
        }
    }
    return 1;
}
```

Output:



```
Enter the lower and upper limit to list primes: 1
1000
All prime number between 1 to 1000 are: 1, 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89,
 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233,
 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397
, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 57
1, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 709, 719, 727, 733, 739, 7
43, 751, 757, 761, 769, 773, 787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863, 877, 881, 883, 887, 907, 911, 919, 929,
937, 941, 947, 953, 967, 971, 977, 983, 991, 997,
```

7.Write a C program to print all strong numbers between given interval using functions.

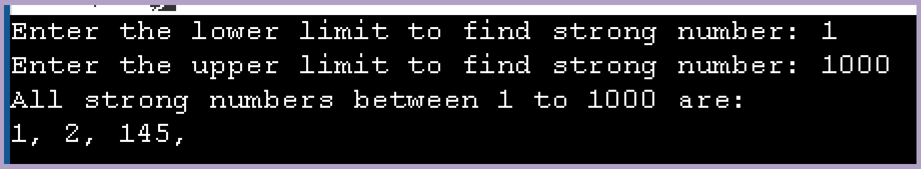
```
#include <stdio.h>


/* Function declaration */
long long fact(int num);
void printStrongNumbers(int start, int end);



int main()
```

```
{
    int start, end;


    /* Input start and end range */
printf("Enter the lower limit to find strong number: ");
scanf("%d", &start);
printf("Enter the upper limit to find strong number: ");
scanf("%d", &end);


printf("All strong numbers between %d to %d are: \n", 
start, end);
printStrongNumbers(start, end);


    return 0;
}
```

```
/**
 * Print all strong numbers in a given range
 */
void printStrongNumbers(int start, int end)
{
    long long sum;
    int num;

    // Iterates from start to end
    while(start != end)
    {
        sum = 0;
num = start;

        // Calculate sum of factorial of digits
        while(num != 0)
        {
            sum += fact(num % 10);
num /= 10;
        }

        // If sum of factorial of digits equal to current
number
        if(start == sum)
        {
printf("%d, ", start);
        }

        start++;
    }
}
```

```
/**
 * Recursively find factorial of any number
 */
long long fact(int num)
{
    if(num == 0)
        return 1;
    else
        return (num * fact(num-1));
}
```

Output:



```
Enter the lower limit to find strong number: 1
Enter the upper limit to find strong number: 1000
All strong numbers between 1 to 1000 are:
1, 2, 145,
```

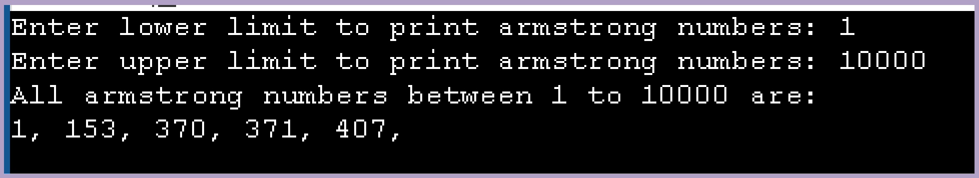
8.Write a C program to print all Armstrong numbers between given interval using functions.

```
#include <stdio.h>
```

```
/* Function declarations */

int isArmstrong(int num);

void printArmstrong(int start, int end);




int main()

{

    int start, end;


    /* Input lower and upper limit to of armstrong
numbers */

printf("Enter lower limit to print armstrong numbers: ");

scanf("%d", &start);

printf("Enter upper limit to print armstrong numbers: ");

scanf("%d", &end);
```

```
printf("All armstrong numbers between %d to %d are:
\n", start, end);

printArmstrong(start, end);


    return 0;

}




/**

 * Check whether the given number is armstrong number
or not.

 * Returns 1 if the number is armstrong otherwise 0.

 */

int isArmstrong(int num)

{

    int temp, lastDigit, sum;
    temp = num;

    sum = 0;


    /* Calculate sum of cube of digits */

    while(temp != 0)

    {

lastDigit = temp % 10;

        sum += lastDigit * lastDigit * lastDigit;

        temp /= 10;

    }


    /*

     * Check if sum of cube of digits equals

     * to original number.

     */

    if(num == sum)

        return 1;
    else
        return 0;
}



/**
 * Print all armstrong numbers between start and end.
 */
void printArmstrong(int start, int end)
{
    /*
     * Iterates from start to end and print the current
number
     * if it is armstrong
     */
    while(start <= end)
    {
        if(isArmstrong(start))

        {

printf("%d, ", start);

        }



        start++;

    }

}
```

Output:



```
Enter lower limit to print armstrong numbers: 1
Enter upper limit to print armstrong numbers: 10000
All armstrong numbers between 1 to 10000 are:
1, 153, 370, 371, 407,
```

9.Write a C program to print all perfect numbers between given interval using functions.

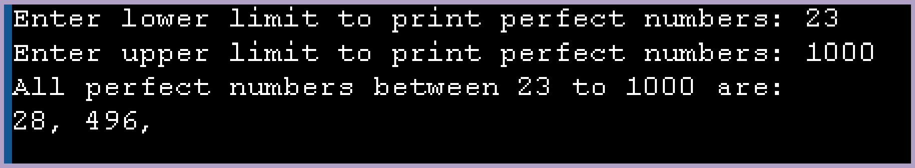
```
#include <stdio.h>



/* Function declarations */
```

```
int isPerfect(int num);

void printPerfect(int start, int end);




int main()

{

    int start, end;


    /* Input lower and upper limit to print perfect
numbers */

printf("Enter lower limit to print perfect numbers: ");

scanf("%d", &start);

printf("Enter upper limit to print perfect numbers: ");

scanf("%d", &end);
```

```
printf("All perfect numbers between %d to %d are: \n",
start, end);
printPerfect(start, end);


    return 0;
}




/**
 * Check whether the given number is perfect or not.
 * Returns 1 if the number is perfect otherwise 0.
 */
int isPerfect(int num)
{
    int i, sum;
    /* Finds sum of all proper divisors */
    sum = 0;
    for(i=1; i<num; i++)
    {
        if(num % i == 0)
        {
            sum += i;
        }
    }

    /*
     * If sum of proper positive divisors equals to given
number
     * then the number is perfect number
     */
    if(sum == num)
        return 1;
    else
        return 0;
}




/**
 * Print all perfect numbers between given range start
and end.
 */
void printPerfect(int start, int end)
{
    /* Iterates from start to end */
    while(start <= end)
    {
        if(isPerfect(start))
        {
```

```
printf("%d, ", start);
        }


        start++;
    }
}
```

Output:



```
Enter lower limit to print perfect numbers: 23
Enter upper limit to print perfect numbers: 1000
All perfect numbers between 23 to 1000 are:
28, 496,
```

10.Write a C program to find power of any number using recursion.

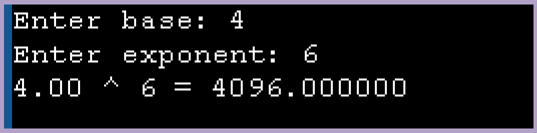
```
#include <stdio.h>



/* Power function declaration */
double pow(double base, int expo);
```

```
int main()
{
    double base, power;
    int expo;

    /* Input base and exponent from user */
printf("Enter base: ");
scanf("%lf", &base);
printf("Enter exponent: ");
scanf("%d", &expo);

    // Call pow function
    power = pow(base, expo);

printf("%.2lf ^ %d = %f", base, expo, power);
```

```
    return 0;
}


/**
 * Calculate power of any number.
 * Returns base ^ expo
 */
double pow(double base, int expo)
{
    /* Base condition */
    if(expo == 0)
        return 1;
    else if(expo > 0)
        return base * pow(base, expo - 1);
    else
```

```
        return 1 / pow(base, -expo);
}
```

Output:



```
Enter base: 4
Enter exponent: 6
4.00 ^ 6 = 4096.000000
```

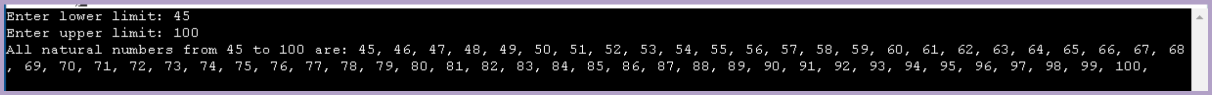
11.Write a C program to print all natural numbers between 1 to n using recursion.

```
#include <stdio.h>



/* Function declaration */
void printNaturalNumbers(int lowerLimit, int upperLimit);



int main()
```

```
{
    int lowerLimit, upperLimit;


    /* Input lower and upper limit from user */
printf("Enter lower limit: ");
scanf("%d", &lowerLimit);
printf("Enter upper limit: ");
scanf("%d", &upperLimit);


printf("All natural numbers from %d to %d are: ",
lowerLimit, upperLimit);
printNaturalNumbers(lowerLimit, upperLimit);


    return 0;
}
```

```
/**
 * Recursively prints all natural number between the
given range.
 */
void printNaturalNumbers(int lowerLimit, int upperLimit)
{
    if(lowerLimit>upperLimit)
        return;

printf("%d, ", lowerLimit);

    // Recursively call the function to print next number
printNaturalNumbers(lowerLimit + 1, upperLimit);
}
```

Output:



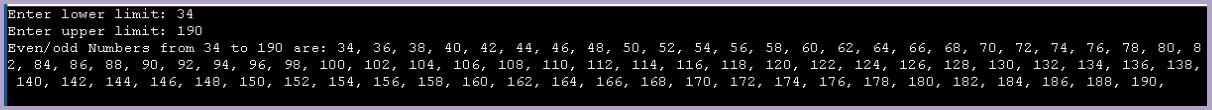
```
Enter lower limit: 45
Enter upper limit: 100
All natural numbers from 45 to 100 are: 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68
, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100,
```

12.Write a C program to print all even or odd numbers in given range using recursion.

```
#include <stdio.h>



/* Function declaration */
void printEvenOdd(int cur, int limit);




int main()
{
    int lowerLimit, upperLimit;

    // Input lower and upper limit from user
printf("Enter lower limit: ");
scanf("%d", &lowerLimit);
```

```
printf("Enter upper limit: ");
scanf("%d", &upperLimit);


printf("Even/odd Numbers from %d to %d are: ",
lowerLimit, upperLimit);
printEvenOdd(lowerLimit, upperLimit);


    return 0;
}




/**
 * Recursive function to print even or odd numbers in a
given range.
 */
void printEvenOdd(int cur, int limit)
{
    if(cur > limit)
        return;


printf("%d, ", cur);


    // Recursively call to printEvenOdd to get next value
printEvenOdd(cur + 2, limit);
}
```

Output:



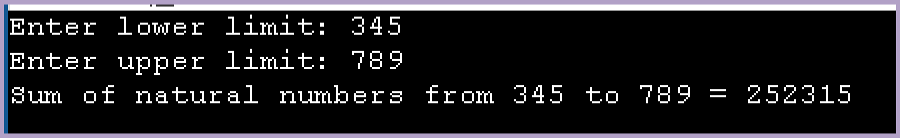
```
Enter lower limit: 34
Enter upper limit: 190
Even/odd Numbers from 34 to 190 are: 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 8
2, 84, 86, 88, 90, 92, 94, 96, 98, 100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138,
 140, 142, 144, 146, 148, 150, 152, 154, 156, 158, 160, 162, 164, 166, 168, 170, 172, 174, 176, 178, 180, 182, 184, 186, 188, 190,
```

13.Write a C program to find sum of all natural numbers between 1 to n using recursion.

```
#include <stdio.h>



/* Function declaration */
int sumOfNaturalNumbers(int start, int end);
int main()

{

    int start, end, sum;


    /* Input lower and upper limit from user */

printf("Enter lower limit: ");

scanf("%d", &start);

printf("Enter upper limit: ");

scanf("%d", &end);


    sum = sumOfNaturalNumbers(start, end);


printf("Sum of natural numbers from %d to %d = %d",
start, end, sum);
```

```
    return 0;
}



/**
 * Recursively find the sum of natural number
 */
int sumOfNaturalNumbers(int start, int end)
{
    if(start == end)
        return start;
    else
        return start + sumOfNaturalNumbers(start + 1, end);
}
```

Output:



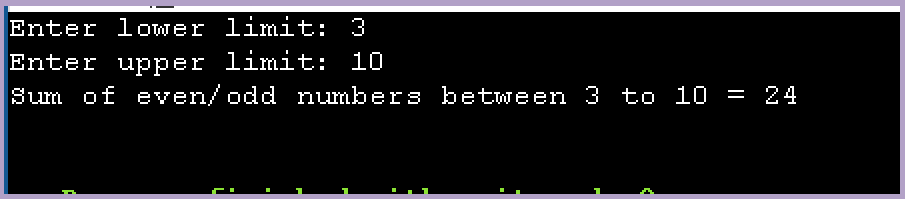
```
Enter lower limit: 345
Enter upper limit: 789
Sum of natural numbers from 345 to 789 = 252315
```

14.Write a C program to find sum of all even or odd numbers in given range using recursion.

```
#include <stdio.h>



int sumOfEvenOdd(int start, int end);



int main()
{
    int start, end, sum;

    /* Input lower and upper limit from user */
printf("Enter lower limit: ");
scanf("%d", &start);
printf("Enter upper limit: ");
scanf("%d", &end);
```

```
printf("Sum of even/odd numbers between %d to %d =
%d\n", start, end, sumOfEvenOdd(start, end));


    return 0;
}





/**
 * Find sum of all even or odd numbers recursively.
 */
int sumOfEvenOdd(int start, int end)
{
    /* Base condition */
    if(start > end)
        return 0;
    else
        return (start + sumOfEvenOdd(start + 2, end));
}
```

Output:



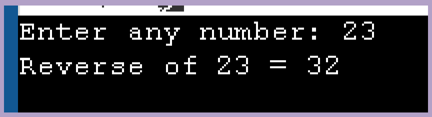
```
Enter lower limit: 3
Enter upper limit: 10
Sum of even/odd numbers between 3 to 10 = 24
```

15.Write a C program to find reverse of any number using recursion.

```
#include <stdio.h>

#include <math.h>



/* Fuction declaration */

int reverse(int num);
int main()
{
    int num, rev;

    /* Input number from user */
printf("Enter any number: ");
scanf("%d", &num);

    /* Call the function to reverse number */
    rev = reverse(num);

printf("Reverse of %d = %d", num, rev);

    return 0;
}
```

```
/**
 * Recursive function to find reverse of any number
 */
int reverse(int num)
{
    // Find total digits in num
    int digit = (int) log10(num);

    // Base condition
    if(num == 0)
        return 0;

    return ((num%10 * pow(10, digit)) + reverse(num/10));
}
```

Output:



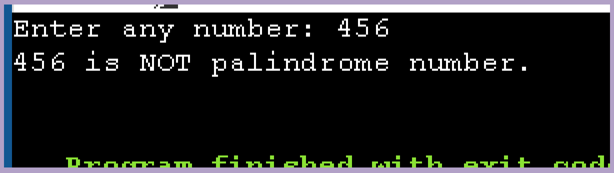
```
Enter any number: 23
Reverse of 23 = 32
```

16.Write a C program to check whether a number is palindrome or not using recursion.

```
#include <stdio.h>

#include <math.h>



/* Function declarations */

int reverse(int num);

int isPalindrome(int num);




int main()

{

    int num;
    /* Input any number from user */
printf("Enter any number: ");
scanf("%d", &num);


    if(isPalindrome(num) == 1)
    {
printf("%d is palindrome number.\n", num);
    }
    else
    {
printf("%d is NOT palindrome number.\n", num);
    }


    return 0;
}
```

```
/**
 * Function to check whether a number is palindrome or
not.
 * This function returns 1 if the number is palindrome
otherwise 0.
 */
int isPalindrome(int num)
{
    /*
     * Check if the given number is equal to
     * its reverse.
     */
    if(num == reverse(num))
    {
        return 1;
    }
    return 0;
}



/**
 * Recursive function to find reverse of any number
 */
int reverse(int num)
{
    /* Find number of digits in num */
    int digit = (int)log10(num);

    /* Recursion base condition */
    if(num == 0)
        return 0;
    return ((num%10 * pow(10, digit)) + reverse(num/10));
}
```

Output:



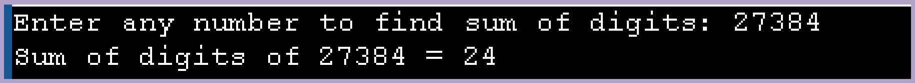
```
Enter any number: 456
456 is NOT palindrome number.


   Program finished with exit cod
```

17.Write a C program to find sum of digits of a given number using recursion.

```
#include <stdio.h>


/* Function declaration */
int sumOfDigits(int num);




int main()
{
    int num, sum;
```

```
printf("Enter any number to find sum of digits: ");
scanf("%d", &num);

    sum = sumOfDigits(num);

printf("Sum of digits of %d = %d", num, sum);

    return 0;
}



/**
 * Recursive function to find sum of digits of a number
 */
int sumOfDigits(int num)
{
    // Base condition
    if(num == 0)
        return 0;

    return ((num % 10) + sumOfDigits(num / 10));
}
```

Output:



```
Enter any number to find sum of digits: 27384
Sum of digits of 27384 = 24
```

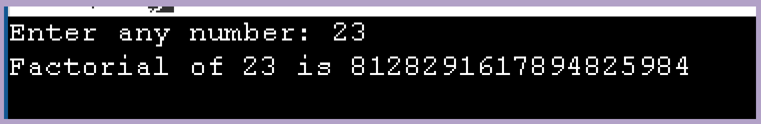
18.Write a C program to find factorial of any number using recursion.

```
#include <stdio.h>

/* Function declaration */
unsigned long long fact(int num);
```

```
int main()
{
    int num;
    unsigned long long factorial;

    /* Input an integer from user */
printf("Enter any number: ");
scanf("%d", &num);

    factorial = fact(num); // Call factorial function

printf("Factorial of %d is %llu", num, factorial);

    return 0;
}
```

```
/**
 * Function to compute and return factorial of any
number recursively.
 */
unsigned long long fact(int num)
{
    // Base condition
    if(num == 0)
        return 1;
    else
        return num * fact(num - 1);
}
```

Output:



```
Enter any number: 23
Factorial of 23 is 8128291617894825984
```

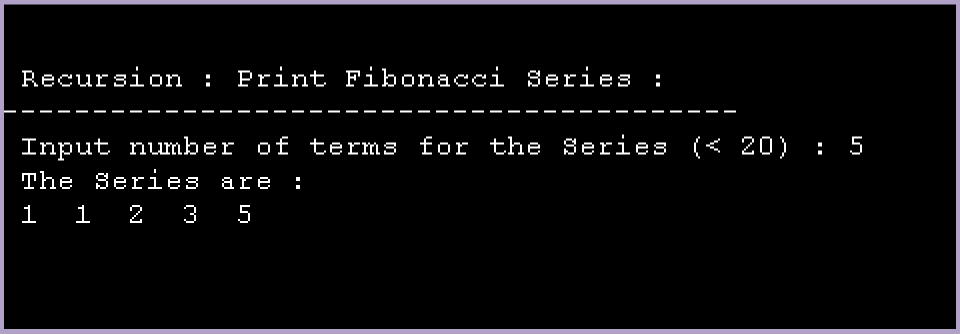
19.Write a C program to generate nth Fibonacci term using recursion.

```
#include <stdio.h>



/* Function declaration */
unsigned long longfibo(int num);



int main()
{
    int num;
    unsigned long longfibonacci;

    /* Input a number from user */
printf("Enter any number to find nth fiboacci term: ");
scanf("%d", &num);

fibonacci = fibo(num);
```

```
printf("%d fibonacci term is %llu", num, fibonacci);


    return 0;
}



/**
 * Recursive function to find nth Fibonacci term
 */
unsigned long longfibo(int num)
{
    if(num == 0)      //Base condition
        return 0;
    else if(num == 1) //Base condition
        return 1;
    else
```

```c
        return fibo(num-1) + fibo(num-2);
}
```

Output



```
Recursion : Print Fibonacci Series :
---------------------------------------
Input number of terms for the Series (< 20) : 5
The Series are :
1  1  2  3  5
```

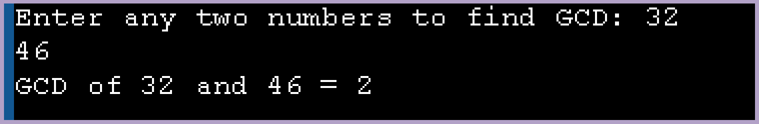
20.Write a C program to find GCD (HCF) of two numbers using recursion.

```c
#include <stdio.h>


/* Function declaration */
int gcd(int a, int b);



int main()
```

```
{
    int num1, num2, hcf;


    /* Input two numbers from user */
printf("Enter any two numbers to find GCD: ");
scanf("%d%d", &num1, &num2);


hcf = gcd(num1, num2);


printf("GCD of %d and %d = %d", num1, num2, hcf);


    return 0;
}



/**
```

```
 * Recursive approach of euclidean algorithm to find GCD of two numbers
 */
int gcd(int a, int b)
{
    if(b == 0)
        return a;
    else
        return gcd(b, a%b);
}
```

Output:



```
Enter any two numbers to find GCD: 32
46
GCD of 32 and 46 = 2
```

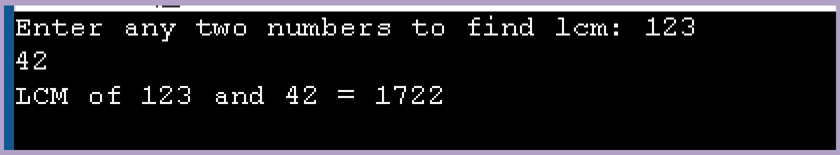
21.Write a C program to find LCM of two numbers using recursion.

```
#include <stdio.h>
```

```
/* Function declaration */
int lcm(int a, int b);



int main()
{
    int num1, num2, LCM;

    /* Input two numbers from user */
printf("Enter any two numbers to find lcm: ");
scanf("%d%d", &num1, &num2);

    /*
     * Ensures that first parameter of LCM function
     * is always less than second
     */
    if(num1 > num2)

        LCM = lcm(num2, num1);

    else

        LCM = lcm(num1, num2);


printf("LCM of %d and %d = %d", num1, num2, LCM);


    return 0;

}




/**

 * Recursive function to find lcm of two numbers 'a' and
'b'.

 * Here 'a' needs to be always less than 'b'.

 */

int lcm(int a, int b)
```

```
{
    static int multiple = 0;


    /* Increments multiple by adding max value to it */
    multiple += b;


    /*
     * Base condition of recursion
     * If found a common multiple then return the
multiple.
     */
    if((multiple % a == 0) && (multiple % b == 0))
    {
        return multiple;
    }
    else
    {
        return lcm(a, b);
    }
}
```

Output:



```
Enter any two numbers to find lcm: 123
42
LCM of 123 and 42 = 1722
```

22.Write a C program to display all array elements using recursion.

```
#include <stdio.h>
#define MAX_SIZE 100

/* Function declaration */
void printArray(int arr[], int start, int len);



int main()
```

```
{
    int arr[MAX_SIZE];
    int N, i;

    /* Input size and elements in array */
printf("Enter size of the array: ");
scanf("%d", &N);
printf("Enter elements in the array: ");
    for(i=0; i<N; i++)
    {
scanf("%d", &arr[i]);
    }

    /* Prints array recursively */
printf("Elements in the array: ");
printArray(arr, 0, N);
```

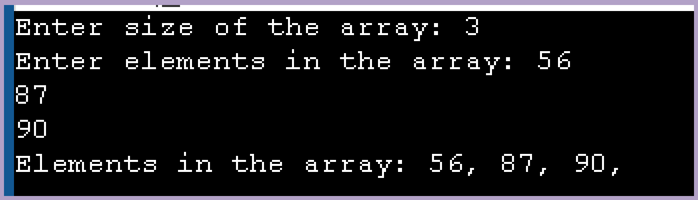
```
    return 0;
}



/**
 * Prints an array recursively within a given range.
 */
void printArray(int arr[], int start, int len)
{
    /* Recursion base condition */
    if(start >= len)
        return;

    /* Prints the current array element */
printf("%d, ", arr[start]);
```

/* Recursively call printArray to print next element in the array */

printArray(arr, start + 1, len);

}

Output:



```
Enter size of the array: 3
Enter elements in the array: 56
87
90
Elements in the array: 56, 87, 90,
```

23.Write a C program to find sum of elements of array using recursion.

#include <stdio.h>

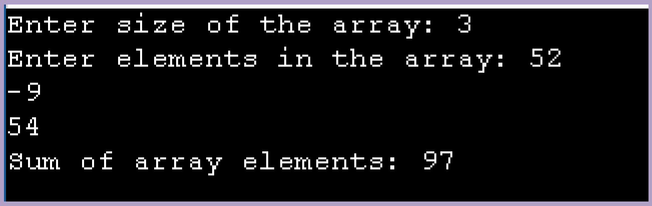
#define MAX_SIZE 100

/* Function declaration to find sum of array */

int sum(int arr[], int start, int len);

```
int main()

{

    int arr[MAX_SIZE];

    int N, i, sumofarray;



    /* Input size and elements in array  */

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

scanf("%d", &N);

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

    for(i=0; i<N; i++)

    {

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

    }



sumofarray = sum(arr, 0, N);
printf("Sum of array elements: %d", sumofarray);


    return 0;
}




/**
 * Recursively find the sum of elements in an array.
 */
int sum(int arr[], int start, int len)
{
    /* Recursion base condition */
    if(start >= len)
        return 0;

    return (arr[start] + sum(arr, start + 1, len));
}
```

Output:


```
Enter size of the array: 3
Enter elements in the array: 52
-9
54
Sum of array elements: 97
```

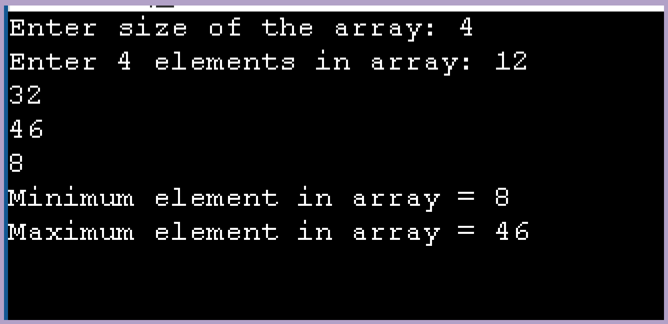
24.Write a C program to find maximum and minimum elements in array using recursion.

```
#include <stdio.h>

#define MAX_SIZE 100 // Maximum size of the array


/* Function declarations */

int maximum(int array[], int index, int len);

int minimum(int array[], int index, int len);




int main()

{

   int array[MAX_SIZE], N, max, min;
```

```
    int i;


    /* Input size and elements of array */
printf("Enter size of the array: ");
scanf("%d", &N);
printf("Enter %d elements in array: ", N);
    for(i=0; i<N; i++)
    {
scanf("%d", &array[i]);
    }


    max = maximum(array, 0, N);
    min = minimum(array, 0, N);


printf("Minimum element in array = %d\n", min);
printf("Maximum element in array = %d\n", max);
```

```
    return 0;
}



/**
 * Recursive function to find maximum element in the
given array.
 */
int maximum(int array[], int index, int len)
{
    int max;

    /*
     * Only last and second last element are left
     */
    if(index >= len-2)
    {
        if(array[index] > array[index + 1])
            return array[index];
        else
            return array[index + 1];
    }


    /*
     * Recursively call maximum to find maximum element
in
     * right side of the array from current index.
     */
    max = maximum(array, index + 1, len);

    /*
     * Compare the current array element with maximum
     * element on its right side
     */
    if(array[index] > max)
        return array[index];
    else
        return max;
}



/**
 * Recursive function to find minimum element in the
array.
 */
int minimum(int array[], int index, int len)
{
    int min;

    if(index >= len-2)
    {
        if(array[index] < array[index + 1])
            return array[index];
        else
            return array[index + 1];
    }

    min = minimum(array, index + 1, len);

    if(array[index] < min)
        return array[index];
    else
        return min;
}
```

Output:



```
Enter size of the array: 4
Enter 4 elements in array: 12
32
46
8
Minimum element in array = 8
Maximum element in array = 46
```

Strings implementation

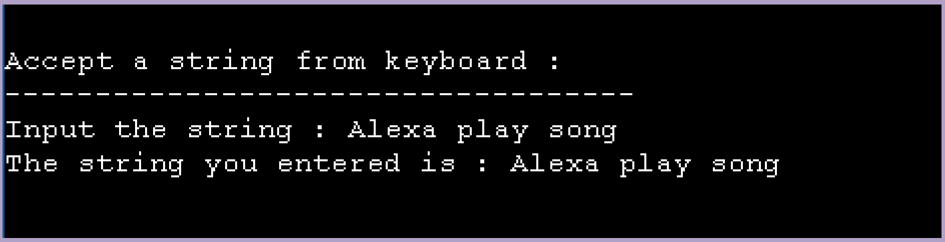
1. Write a program in C to input a string and print it.

Source code:

```
#include <stdio.h>

#include <stdlib.h>


void main()

{

   char str[50];


   printf("\n\nAccept a string from keyboard :\n");
```

```
    printf("----------------------------------\n");
    printf("Input the string : ");
    fgets(str, sizeof str, stdin);
    printf("The string you entered is : %s\n", str);
}
```

output:



```
Accept a string from keyboard :
-----------------------------------
Input the string : Alexa play song
The string you entered is : Alexa play song
```

2. Write a program in C to find the length of a string without using library function.

Test Data :

Input the string : PDEU.com

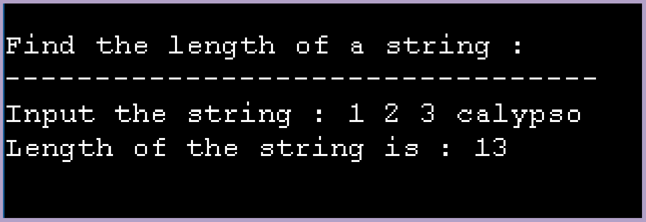
Expected Output :

Length of the string is : 15

Source code:

```
#include <stdio.h>

#include <stdlib.h>


void main()

{

    char str[100]; /* Declares a string of size 100 */

    int l= 0;


       printf("\n\nFind the length of a string :\n");

       printf("--------------------------------\n");

       printf("Input the string : ");

       fgets(str, sizeof str, stdin);

    while(str[l]!='\0')

    {

        l++;

    }
    printf("Length of the string is : %d\n\n", l-1);
}
```

output:



```
Find the length of a string :
---------------------------------
Input the string : 1 2 3 calypso
Length of the string is : 13
```

3. Write a program in C to separate the individual characters from a string.

Test Data :

Input the string : PDEU.com

Expected Output :

The characters of the string are :

P D E U . C O M

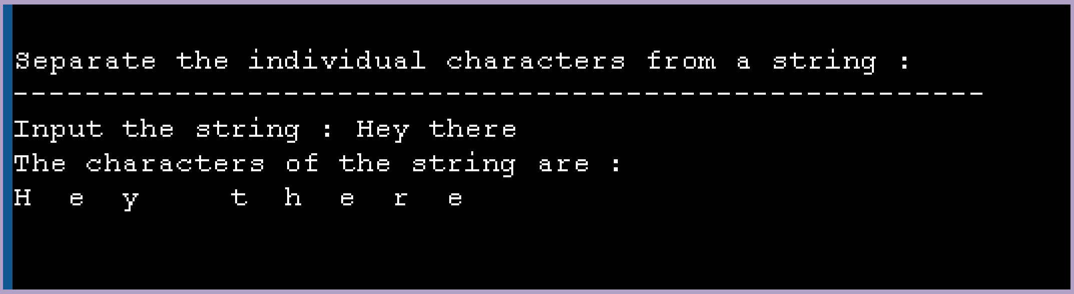
Source code:

```
#include <stdio.h>
```

```
#include <stdlib.h>



void main()
{
    char str[100]; /* Declares a string of size 100 */
    int l= 0;

       printf("\n\nSeparate the individual characters from a
string :\n");
       printf("-----------------------------------------------------\n");
       printf("Input the string : ");
       fgets(str, sizeof str, stdin);
           printf("The characters of the string are : \n");
    while(str[l]!='\0')
    {
       printf("%c  ", str[l]);
```

```
        l++;
    }
    printf("\n");
}
```

output:



```
Separate the individual characters from a string :
------------------------------------------------------
Input the string : Hey there
The characters of the string are :
H  e  y     t  h  e  r  e
```

4. Write a program in C to print individual characters of string in reverse order.

Test Data :

Input the string : PDEU.com

Expected Output :

The characters of the string in reverse are :

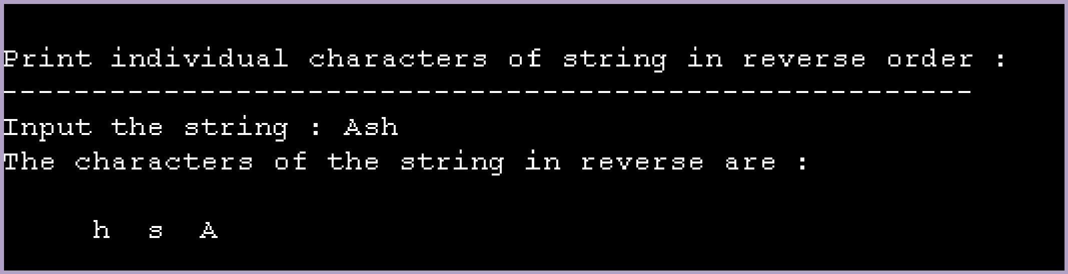
MOC.UEDP

Source code:

```
#include <stdio.h>

#include <string.h>

#include <stdlib.h>


void main()

{

    char str[100]; /* Declares a string of size 100 */

    int l,i;


       printf("\n\nPrint individual characters of string in
reverse order :\n");

       printf("------------------------------------------------------\n");

       printf("Input the string : ");

       fgets(str, sizeof str, stdin);

           l=strlen(str);
```

```
          printf("The characters of the string in reverse are :
\n");
      for(i=l;i>=0;i--)
       {
         printf("%c  ", str[i]);
       }
   printf("\n");
}
```

output:



```
Print individual characters of string in reverse order :
------------------------------------------------------
Input the string : Ash
The characters of the string in reverse are :

     h  s  A
```

5. Write a program in C to count the total number of words in a string.

Test Data :

Input the string : This is PDEU.com

Expected Output :

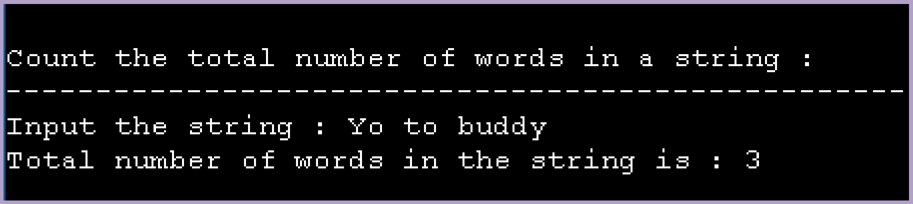
```
Total number of words in the string is : 1
```

Source code:

```
#include <stdio.h>

#include <string.h>

#include <stdlib.h>


#define str_size 100 //Declare the maximum size of the
string


void main()

{

    char str[str_size];

    int i, wrd;


       printf("\n\nCount the total number of words in a
string :\n");
```

```
        printf("----------------------------------------------------\n");
        printf("Input the string : ");
        fgets(str, sizeof str, stdin);


    i = 0;
    wrd = 1;


    /* loop till end of string */
    while(str[i]!='\0')
    {
        /* check whether the current character is white
space or new line or tab character*/
        if(str[i]==' ' || str[i]=='\n' || str[i]=='\t')
        {
            wrd++;
        }
        i++;

    }


    printf("Total number of words in the string is : %d\n",
wrd-1);

}
```

output:



```
Count the total number of words in a string :
------------------------------------------------
Input the string : Yo to buddy
Total number of words in the string is : 3
```

6. Write a program in C to compare two strings without using string library functions.

Test Data :

Check the length of two strings:

Input the 1st string : aabbcc

Input the 2nd string : abcdef

String1: aabbcc

String2: abcdef

Expected Output : Strings are not equal.

Check the length of two strings:

Input the 1st string : aabbcc

Input the 2nd string : aabbcc

String1: aabbcc

String2: aabbcc

Expected Output : Strings are equal.

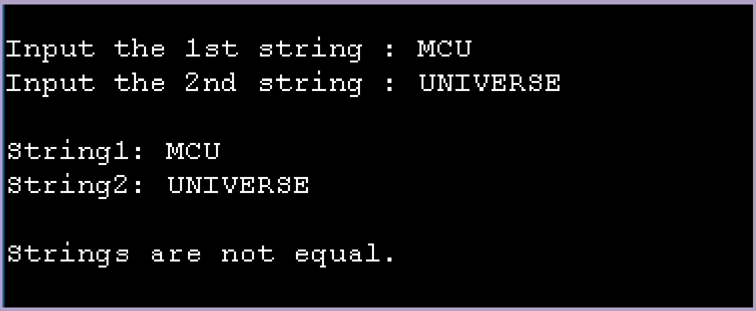
Source code:

```
#include <stdio.h>
```

```
#define str_size 100 //Declare the maximum size of the
string

int test(char* s1, char* s2)

{

        int flag = 0;

        while (*s1 != '\0' || *s2 != '\0') {

                if (*s1 == *s2) {

                        s1++;

                        s2++;

                }

       else if ((*s1 == '\0' && *s2 != '\0')

                            || (*s1 != '\0' && *s2 == '\0')

                            || *s1 != *s2) {

                        flag = 1;

                        break;

                }

        }
```

```
  return flag;
}
int main(void)
{
char str1[str_size], str2[str_size];
   int flg=0;
   printf("\nInput the 1st string : ");
   fgets(str1, sizeof str1, stdin);
   printf("Input the 2nd string : ");
   fgets(str2, sizeof str2, stdin);
   printf("\nString1: %s", str1);
   printf("String2: %s", str2);
   flg = test(str1, str2);
    if(flg == 0)
   {
       printf("\nStrings are equal.\n");
   }
   else if(flg == 1)
   {
      printf("\nStrings are not equal.");
   }
         return 0;
}
```

output:



```
Input the 1st string : MCU
Input the 2nd string : UNIVERSE

String1: MCU
String2: UNIVERSE

Strings are not equal.
```

7. Write a program in C to count total number of alphabets, digits and special characters in a string.

Test Data :

Input the string : Welcome to PDEU.com

Expected Output :

```
Number of Alphabets in the string is : 21

Number of Digits in the string is : 0

Number of Special characters in the string is : 1
```

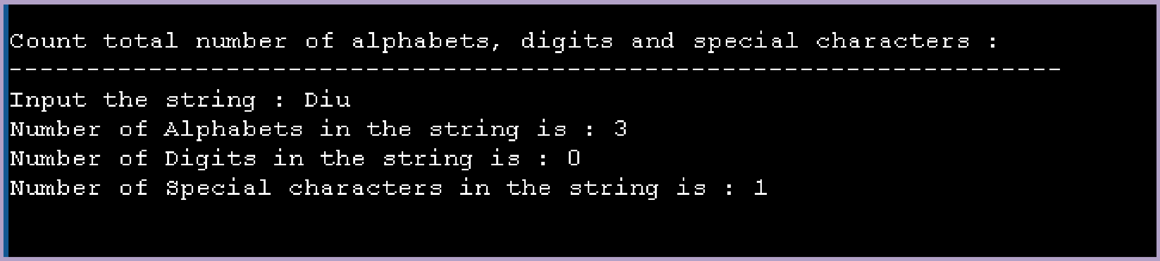
Source code:

```
#include <stdio.h>

#include <string.h>

#include <stdlib.h>



#define str_size 100 //Declare the maximum size of the
string


void main()

{

    char str[str_size];

    int alp, digit, splch, i;
    alp = digit = splch = i = 0;



       printf("\n\nCount total number of alphabets, digits
and special characters :\n");
       printf("---------------------------------------------------
-------\n");
       printf("Input the string : ");
       fgets(str, sizeof str, stdin);

     /* Checks each character of string*/

    while(str[i]!='\0')
    {
        if((str[i]>='a' && str[i]<='z') || (str[i]>='A' &&
str[i]<='Z'))
        {
            alp++;
        }
        else if(str[i]>='0' && str[i]<='9')
        {
            digit++;
        }
        else
        {
            splch++;
        }

        i++;
    }

    printf("Number of Alphabets in the string is : %d\n",
alp);
    printf("Number of Digits in the string is : %d\n", digit);
```

    printf("Number of Special characters in the string is : %d\n\n", splch);

}

output:



```
Count total number of alphabets, digits and special characters :
--------------------------------------------------------------------
Input the string : Diu
Number of Alphabets in the string is : 3
Number of Digits in the string is : 0
Number of Special characters in the string is : 1
```

8. Write a program in C to copy one string to another string.

Test Data :

Input the string : This is a string to be copied.

Expected Output :

The First string is : This is a string to be copied.

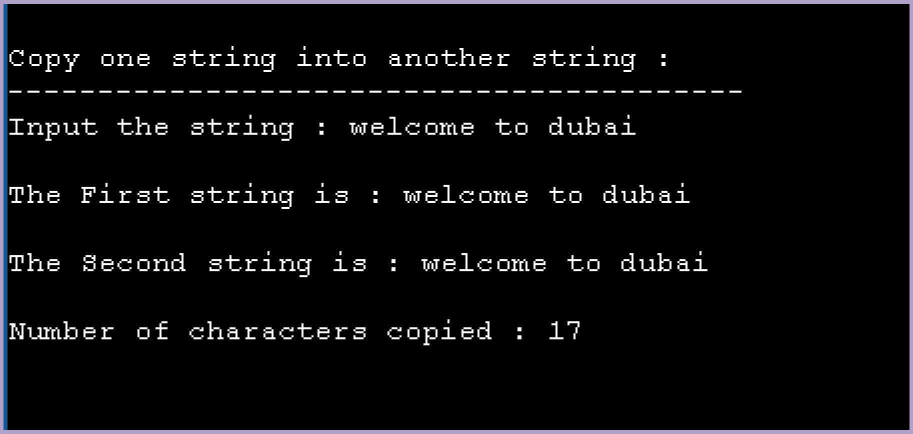
The Second string is : This is a string to be copied.

Number of characters copied : 31

Source code:

```
#include <stdio.h>

#include <string.h>

#include <stdlib.h>



void main()

{

    char str1[100], str2[100];

    int  i;


       printf("\n\nCopy one string into another string :\n");

       printf("-----------------------------------------\n");

       printf("Input the string : ");

       fgets(str1, sizeof str1, stdin);


    /* Copies string1 to string2 character by character */
    i=0;

    while(str1[i]!='\0')

    {

        str2[i] = str1[i];

        i++;

    }


    //Makes sure that the string is NULL terminated

    str2[i] = '\0';


    printf("\nThe First string is : %s\n", str1);

    printf("The Second string is : %s\n", str2);

    printf("Number of characters copied : %d\n\n", i);

}
```

output:



```
Copy one string into another string :
-----------------------------------------
Input the string : welcome to dubai

The First string is : welcome to dubai

The Second string is : welcome to dubai

Number of characters copied : 17
```

9. Write a program in C to count total number of vowel or consonant in a string.

Test Data :

Input the string : Welcome to PDEU.com

Expected Output :

The total number of vowel in the string is : 9

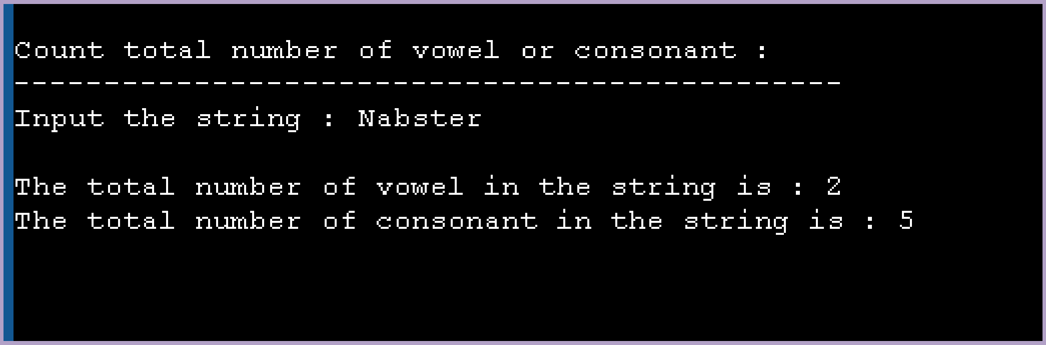
The total number of consonant in the string is : 12

Source code:

```
#include <stdio.h>

#include <string.h>
```

```
#include <stdlib.h>


#define str_size 100 //Declare the maximum size of the
string


void main()
{
    char str[str_size];
    int i, len, vowel, cons;


       printf("\n\nCount total number of vowel or
consonant :\n");
       printf("-------------------------------------------------\n");
       printf("Input the string : ");
       fgets(str, sizeof str, stdin);


    vowel = 0;
    cons = 0;

    len = strlen(str);


    for(i=0; i<len; i++)

    {


        if(str[i] =='a' || str[i]=='e' || str[i]=='i' || str[i]=='o' ||
str[i]=='u' || str[i]=='A' || str[i]=='E' || str[i]=='I' ||
str[i]=='O' || str[i]=='U')

        {

            vowel++;

        }

        else if((str[i]>='a' && str[i]<='z') || (str[i]>='A' &&
str[i]<='Z'))

        {

            cons++;

        }

    }
    printf("\nThe total number of vowel in the string is : %d\n", vowel);
    printf("The total number of consonant in the string is : %d\n\n", cons);
}
```

output:



```
Count total number of vowel or consonant :
----------------------------------------------
Input the string : Nabster

The total number of vowel in the string is : 2
The total number of consonant in the string is : 5
```

10. Write a program in C to find maximum occurring character in a string.

Test Data :

Input the string : Welcome to PDEU.com.

Expected Output :

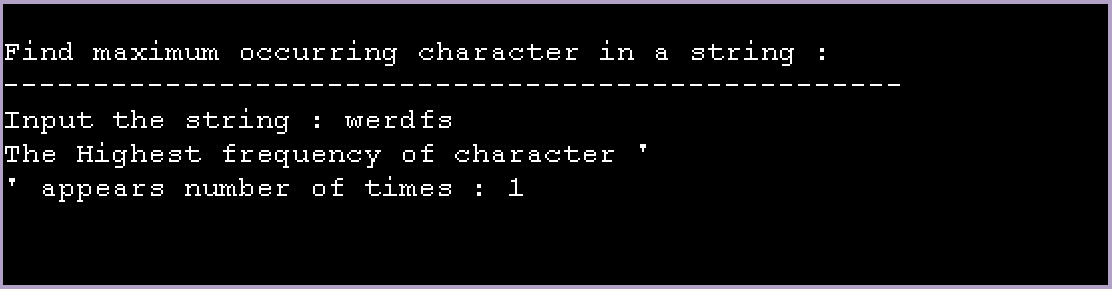
```
The Highest frequency of character 'e'
appears number of times : 4
```

Source code:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define str_size 100 //Declare the maximum size of the
string
#define chr_no 255 //Maximum number of characters to
be allowed

void main()
{
  char str[str_size];
    int ch_fre[chr_no];
  int i = 0, max;

  int ascii;


     printf("\n\nFind maximum occurring character in a
string :\n");

     printf("----------------------------------------------\n");

     printf("Input the string : ");

     fgets(str, sizeof str, stdin);




  for(i=0; i<chr_no; i++)  //Set frequency of all characters
to 0

  {

    ch_fre[i] = 0;

  }
```

```
/* Read for frequency of each characters */

i=0;

while(str[i] != '\0')

{

    ascii = (int)str[i];

    ch_fre[ascii] += 1;


    i++;

}


max = 0;

for(i=0; i<chr_no; i++)

{

    if(i!=32)

    {

    if(ch_fre[i] > ch_fre[max])

        max = i;
        }
    }
    printf("The Highest frequency of character '%c'
appears number of times : %d \n\n", max, ch_fre[max]);
}
```

output



```
Find maximum occurring character in a string :
--------------------------------------------------
Input the string : werdfs
The Highest frequency of character '
' appears number of times : 1
```

11. Write a C program to sort a string array in ascending order.

Test Data :

Input the string : PDEU

Expected Output :

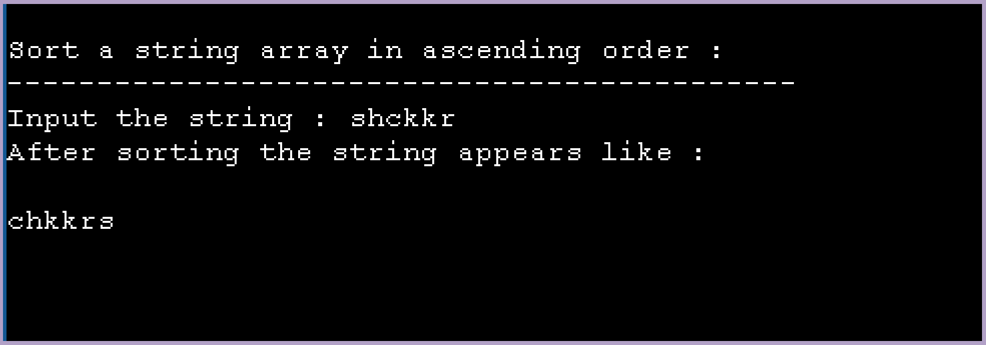
After sorting the string appears like :

Source code:

```
#include <stdio.h>

#include <string.h>

void main()

{

  char str[100],ch;

  int i,j,l;


       printf("\n\nSort a string array in ascending order
:\n");

       printf("--------------------------------------------\n");

       printf("Input the string : ");

       fgets(str, sizeof str, stdin);

  l=strlen(str);

  /* sorting process */

  for(i=1;i<l;i++)

    for(j=0;j<l-i;j++)
```

```
        if(str[j]>str[j+1])
        {
          ch=str[j];
          str[j] = str[j+1];
          str[j+1]=ch;
        }
   printf("After sorting the string appears like : \n");
   printf("%s\n\n",str);
  }
```

output:



```
Sort a string array in ascending order :
--------------------------------------------
Input the string : shckkr
After sorting the string appears like :

chkkrs
```

12. Write a program in C to read a strings through keyboard and sort the words.

Test Data :

Input number of strings :3

Input string 3 :

zero

one

two

Expected Output :

The strings appears after sorting :

one

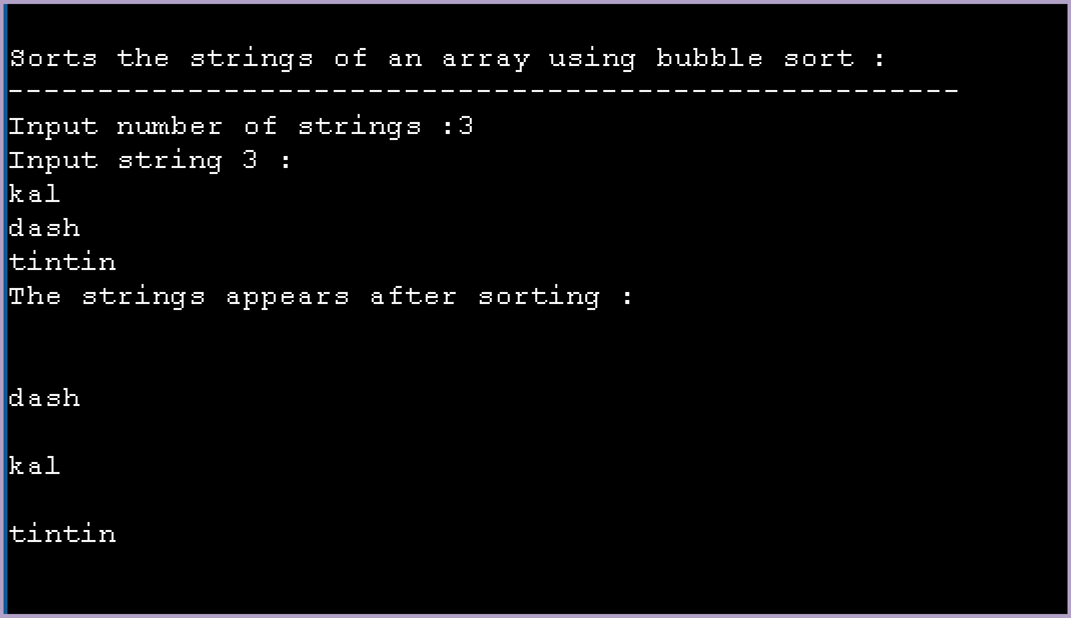
two

zero

Source code:

```
#include <stdio.h>

#include <string.h>

void main()

{
  char name[25][50],temp[25];

  int n,i,j;


      printf("\n\nSorts the strings of an array using bubble
sort :\n");

      printf("--------------------------------------------------\n");




  printf("Input number of strings :");

  scanf("%d",&n);


printf("Input string %d :\n",n);

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

  {


      fgets(name[i], sizeof name, stdin);

  }
```

```
      /*Logic Bubble Sort*/


      for(i=1;i<=n;i++)
         for(j=0;j<=n-i;j++)
           if(strcmp(name[j],name[j+1])>0)
           {
             strcpy(temp,name[j]);
             strcpy(name[j],name[j+1]);
             strcpy(name[j+1],temp);
           }
    printf("The strings appears after sorting :\n");
             for(i=0;i<=n;i++)
                printf("%s\n",name[i]);


 }
 output:
```


```
Sorts the strings of an array using bubble sort :
-----------------------------------------------------
Input number of strings :3
Input string 3 :
kal
dash
tintin
The strings appears after sorting :


dash

kal

tintin
```

13. Write a program in C to extract a substring from a given string.

Test Data :

Input the string : this is test string

Input the position to start extraction :9

Input the length of substring :4

Expected Output :

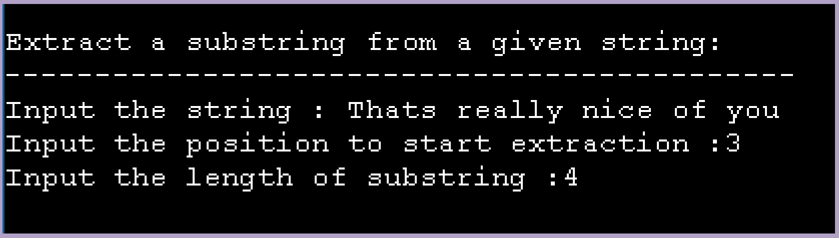
```
The substring retrieve from the string is : " test "
```

Source code:

```
#include <stdio.h>

void main()

{

   char str[100], sstr[100];

   int pos, l, c = 0;


       printf("\n\nExtract a substring from a given
string:\n");

       printf("------------------------------------------\n");


       printf("Input the string : ");

       fgets(str, sizeof str, stdin);


   printf("Input the position to start extraction :");
```

```
  scanf("%d", &pos);


  printf("Input the length of substring :");
  scanf("%d", &l);


  while (c < l)
  {
    sstr[c] = str[pos+c-1];
    c++;
  }
```

output:



```
Extract a substring from a given string:
--------------------------------------------
Input the string : Thats really nice of you
Input the position to start extraction :3
Input the length of substring :4
```

14. Write a C program to check whether a given substring is present in the given string.

Test Data :

Input the string : This is a test string.

Input the substring to be search : search

Expected Output :

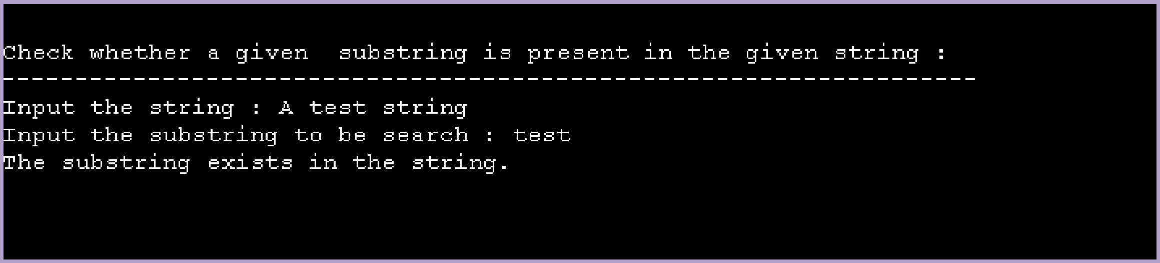
The substring is not exists in the string.

Source code:

```
#include <stdio.h>
void main()
{
	char str[80],search[20];
	int c1=0,c2=0,i,j,flg;
	printf("\n\nCheck whether a given  substring is
present in the given string :\n");
	printf("--------------------------------------------------------------
------\n");



	printf("Input the string : ");
	fgets(str, sizeof str, stdin);


	printf("Input the substring to be search : ");
	fgets(search, sizeof search, stdin);


	while (str[c1]!='\0')
		c1++;
		c1--;


	while (search[c2]!='\0')
```

```
        c2++;

        c2--;



for(i=0;i<=c1-c2;i++)

{

        for(j=i;j<i+c2;j++)

        {

                flg=1;

                if (str[j]!=search[j-i])

                {

                        flg=0;

                   break;

                }

        }

        if (flg==1)

                break;
        }
        if (flg==1)
                printf("The substring exists in the string.\n\n");
        else
                printf("The substring is not exists in the string.
\n\n");
}
```

output:



```
Check whether a given  substring is present in the given string :
-------------------------------------------------------------------
Input the string : A test string
Input the substring to be search : test
The substring exists in the string.
```

15. Write a program in C to read a sentence and replace lowercase characters by uppercase and vice-versa.

Test Data :

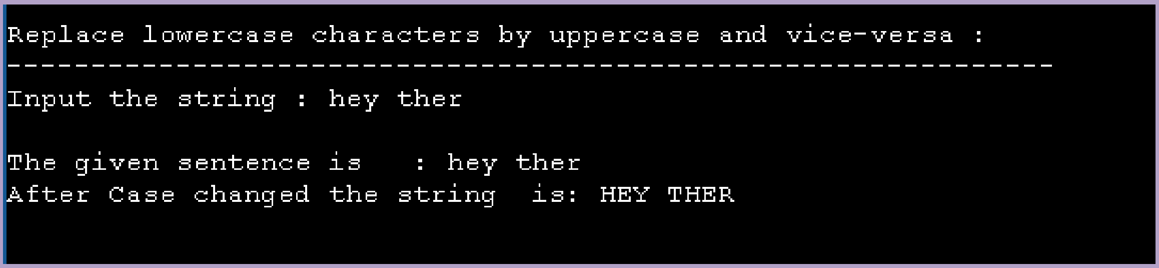
```
Input the string : This Is A Test String.

Expected Output :

The given sentence is   : This Is A Test String.

After Case changed the string  is: tHIS iS a tEST sTRING.
```

Source code:

```
#include <stdio.h>

#include <string.h>

#include <ctype.h>


void main()

{

  char str[100];

  int ctr, ch, i;


      printf("\n\nReplace lowercase characters by
uppercase and vice-versa :\n");
```

```
       printf("-----------------------------------------------------------
-\n");


          printf("Input the string : ");

       fgets(str, sizeof str, stdin);


  i=strlen(str);


  ctr = i; /*shows the number of chars accepted in a
sentence*/


  printf("\nThe given sentence is   : %s",str);


  printf("After Case changed the string  is: ");

  for(i=0; i < ctr; i++)

  {

    ch = islower(str[i]) ? toupper(str[i]) : tolower(str[i]);
    putchar(ch);
   }
   printf("\n\n");


}
```

output:



```
Replace lowercase characters by uppercase and vice-versa :
--------------------------------------------------------------
Input the string : hey ther

The given sentence is    : hey ther
After Case changed the string  is: HEY THER
```

16. Write a program in C to find the number of times a given word 'the' appears in the given string.

Test Data :

Input the string : The string where the word the present more than once.

Expected Output :

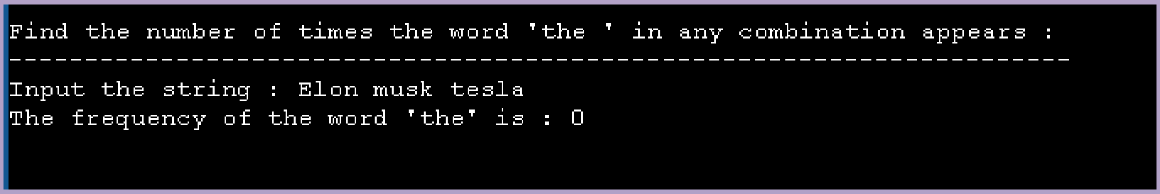
The frequency of the word 'the' is : 3

Source code:

```
#include <stdio.h>

#include <string.h>

void main()

{

        int ctr=0,i,freq=0;

        int t,h,e,spc;

        char str[100];



       printf("\n\nFind the number of times the word 'the '
in any combination appears :\n");

       printf("----------------------------------------------------------------
---------\n");


        printf("Input the string : ");
        fgets(str,sizeof str,stdin);


        ctr=strlen(str);


        /*Counts the frequency of the word 'the' with a
    trailing space*/
        for(i=0;i<=ctr-3;i++)
        {
            t=(str[i]=='t'||str[i]=='T');
            h=(str[i+1]=='h'||str[i+1]=='H');
            e=(str[i+2]=='e'||str[i+2]=='E');
            spc=(str[i+3]==' '||str[i+3]=='\0');
            if ((t&&h&&e&&spc)==1)
              freq++;
        }
        printf("The frequency of the word \'the\' is :
    %d\n\n",freq);
```

}

output:



```
Find the number of times the word 'the ' in any combination appears :
----------------------------------------------------------------------
Input the string : Elon musk tesla
The frequency of the word 'the' is : 0
```

17. Write a program in C to remove characters in String Except Alphabets.

Test Data :

Input the string : PDEU.com

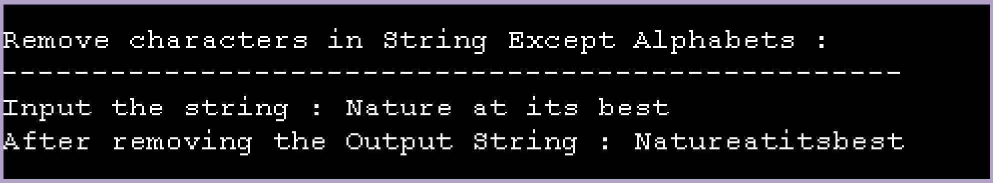
Expected Output :

After removing the Output String : PDEUcom

Source code:

```
#include <stdio.h>

#include <string.h>

void main(){
    char str[150];

    int i,j;


       printf("\n\nRemove characters in String Except
Alphabets :\n");

       printf("--------------------------------------------------\n");


        printf("Input the string : ");

    fgets(str,sizeof str,stdin);

    for(i=0; str[i]!='\0'; ++i)

    {

        while (!((str[i]>='a'&&str[i]<='z') ||
(str[i]>='A'&&str[i]<='Z' || str[i]=='\0')))

        {

            for(j=i;str[j]!='\0';++j)

            {

                str[j]=str[j+1];
            }
            str[j]='\0';
        }
    }
    printf("After removing the Output String : %s\n\n",str);
}
```

output:



```
Remove characters in String Except Alphabets :
--------------------------------------------------
Input the string : Nature at its best
After removing the Output String : Natureatitsbest
```

18. Write a program in C to Find the Frequency of Characters.

Test Data :

Input the string : This is a test string

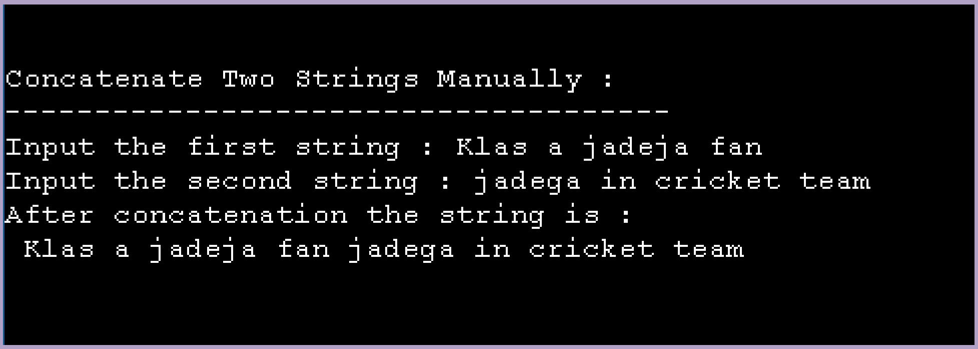
Input the character to find frequency: i

Expected Output :

The frequency of 'i' is : 3

```
#include <stdio.h>

#include <string.h>

void main()

{

    char str1[100], str2[100], i, j,l,m,k;


       printf("\n\nConcatenate Two Strings Manually :\n");

       printf("-------------------------------------\n");


        printf("Input the first string : ");

    fgets(str1,sizeof str1,stdin);

        printf("Input the second string : ");

    fgets(str2,sizeof str2,stdin);

    l=strlen(str1);
    m=strlen(str2);

    for(i=0; i<l-1; ++i);  /* value i contains reaches the end
of string str1. */

    str1[i]=' ';           /* add a space with string str1. */

    i++;                  /* value i increase by 1 for the blank
space */


    for(j=0; j<m-1; ++j, ++i)

    {

        str1[i]=str2[j];

    }

     k=strlen(str1);


    printf("After concatenation the string is : \n ");

    for(i=0; i<k; ++i)

    {

         printf("%c",str1[i]);
    }
printf("\n\n");
}
```

output:



```
Concatenate Two Strings Manually :
-------------------------------------
Input the first string : Klas a jadeja fan
Input the second string : jadega in cricket team
After concatenation the string is :
 Klas a jadeja fan jadega in cricket team
```

19. Write a program in C to Concatenate Two Strings Manually.

Test Data :

Input the first string : this is string one

Input the second string : this is string two

Expected Output :

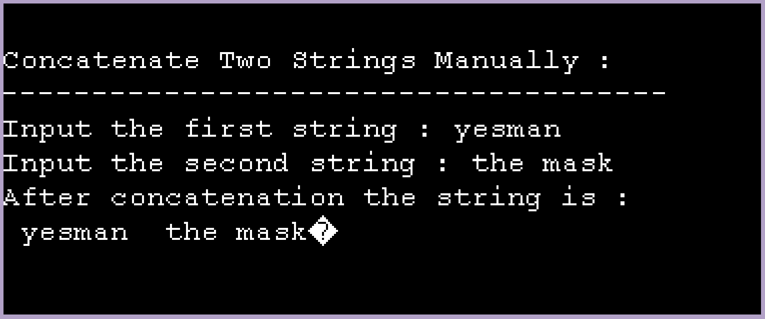
After concatenation the string is :

```
this is string one this is string two
```

Source code:

```
#include <stdio.h>

#include <string.h>

void main()

{

    char str1[100], str2[100], i, j,l,m,k;


       printf("\n\nConcatenate Two Strings Manually :\n");

       printf("-------------------------------------\n");


        printf("Input the first string : ");

    fgets(str1,sizeof str1,stdin);

        printf("Input the second string : ");

    fgets(str2,sizeof str2,stdin);

    l=strlen(str1);
    m=strlen(str2);

    for(i=0; i<l-1; ++i);  /* value i contains reaches the end
of string str1. */

    str1[i]=' ';           /* add a space with string str1. */

    i++;                   /* value i increase by 1 for the blank
space */


    for(j=0; j<m-1; ++j, ++i)

    {

        str1[i]=str2[j];

    }

     k=strlen(str1);


    printf("After concatenation the string is : \n ");

    for(i=0; i<k; ++i)

    {

         printf("%c",str1[i]);
    }
printf("\n\n");
}
```

output:



```
Concatenate Two Strings Manually :
-------------------------------------
Input the first string : yesman
Input the second string : the mask
After concatenation the string is :
 yesman  the mask�
```

20. Write a program in C to find the largest and smallest word in a string.

Test Data :

Input the string : It is a string with smallest and largest word.

Expected Output :

The largest word is 'smallest'

and the smallest word is 'a'

in the string : 'It is a string with smallest and largest word.'.

Source code:

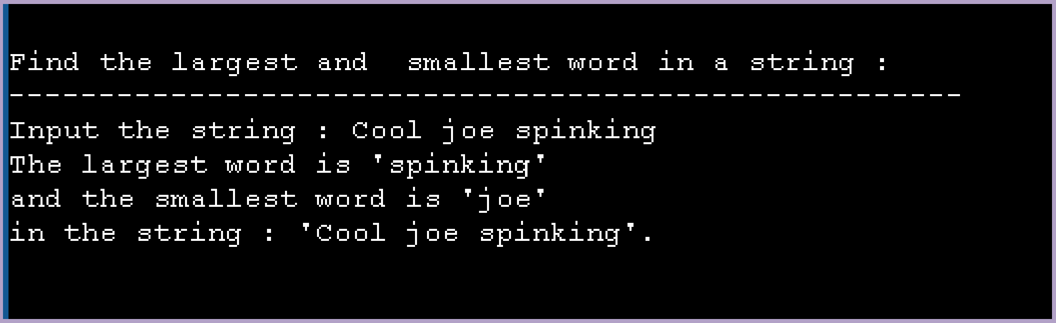
```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

void main()
{
   char str[100], word[20], mx[20], mn[20], c;
   int i = 0, j = 0, flg = 0;

      printf("\n\nFind the largest and  smallest word in a string :\n");
      printf("--------------------------------------------------------\n");
```

```
        printf("Input the string : ");
    i = 0;
    do
    {
        fflush(stdin);
        c = getchar();
        str[i++] = c;

    } while (c != '\n');
    str[i - 1] = '\0';
    for (i = 0; i < strlen(str); i++)
    {
        while (i < strlen(str) && !isspace(str[i]) &&
isalnum(str[i]))
        {
            word[j++] = str[i++];
        }
```

```
if (j != 0)
{
    word[j] = '\0';
    if (!flg)
    {
        flg = !flg;
        strcpy(mx, word);
        strcpy(mn, word);
    }
    if (strlen(word) > strlen(mx))
    {
        strcpy(mx, word);
    }
    if (strlen(word) < strlen(mn))
    {
        strcpy(mn, word);
    }
```

```
            j = 0;
        }
    }
    printf("The largest word is '%s' \nand the smallest
word is '%s' \nin the string : '%s'.\n", mx, mn, str);
 }
```

output:



```
Find the largest and  smallest word in a string :
-----------------------------------------------------
Input the string : Cool joe spinking
The largest word is 'spinking'
and the smallest word is 'joe'
in the string : 'Cool joe spinking'.
```

21. Write a program in C to convert a string to uppercase.

Test Data :

Input a string in lowercase : the quick brown fox jumps over the lazy dog

Expected Output :

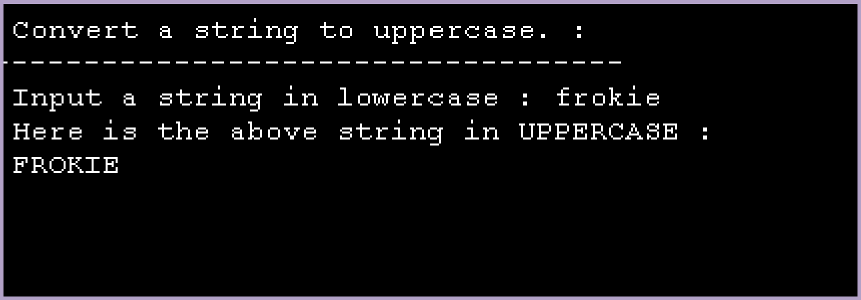
Here is the above string in UPPERCASE :

 THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG.

Source code:

```
#include<stdio.h>
#include<ctype.h>

int main()
{
        int ctr=0;
        char str_char;
        char str[100];

        printf("\n Convert a string to uppercase. :\n");
    printf("----------------------------------");
    printf("\n Input a string in lowercase : ");
    fgets(str, sizeof str, stdin);
    printf(" Here is the above string in UPPERCASE :\n ");
       while (str[ctr])
       {
             str_char=str[ctr];
             putchar (toupper(str_char));
             ctr++;
       }
       printf("\n\n");
       return 0;
}
```

output:



```
Convert a string to uppercase. :
-----------------------------------
Input a string in lowercase : frokie
Here is the above string in UPPERCASE :
FROKIE
```

22. Write a program in C to convert a string to lowercase.

Test Data :

Input a string in UPPERCASE : THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG.

Expected Output :

Here is the above string in lowercase :

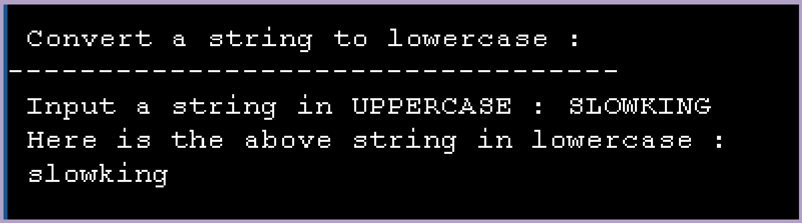
the quick brown fox jumps over the lazy dog.

Source code:

```
#include<stdio.h>

#include<ctype.h>


int main()

{

        int ctr=0;

        char str_char;

        char str[100];
```

```
    printf("\n Convert a string to lowercase :\n");
  printf("------------------------------------");
  printf("\n Input a string in UPPERCASE : ");
  fgets(str, sizeof str, stdin);
  printf(" Here is the above string in lowercase :\n ");
    while (str[ctr])
    {
        str_char=str[ctr];
        putchar (tolower(str_char));
        ctr++;
    }
    return 0;
}
```

output:



```
Convert a string to lowercase :
-----------------------------------
Input a string in UPPERCASE : SLOWKING
Here is the above string in lowercase :
slowking
```

24. Write a program in C to check whether a letter is uppercase or not.

Test Data :

Input a character : p

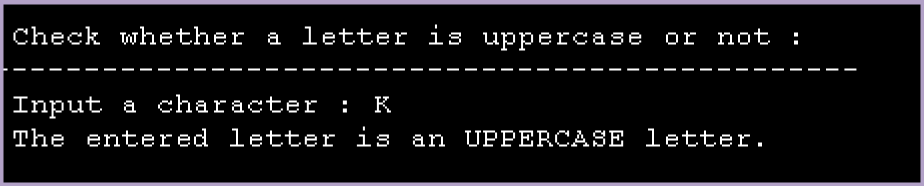
Expected Output :

The entered letter is not an UPPERCASE letter.

Source code:

```
#include<stdio.h>

#include<ctype.h>


int main()

{

        char TestChar;
        printf("\n Check whether a letter is uppercase or not
:\n");
    printf("----------------------------------------------\n");
    printf(" Input a character : ");
    scanf( "%c", &TestChar );
        if( isupper(TestChar) )
                printf( " The entered letter is an UPPERCASE
letter. \n" );
        else
                printf( " The entered letter is not an UPPERCASE
letter. \n" );
        return 0;
}
```

output:



```
Check whether a letter is uppercase or not :
------------------------------------------------
Input a character : K
The entered letter is an UPPERCASE letter.
```

25. Write a program in C to replace the spaces of a string with a specific character.

Test Data :

Input a string : Be glad to see the back of Input replace character : *

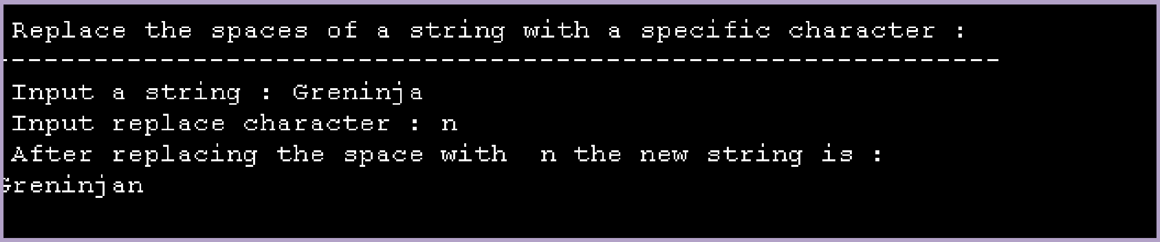
Expected Output :

```
After replacing the space with  * the new string is :

Be*glad*to*see*the*back*of*
```

Source code:

```
#include<stdio.h>

#include<ctype.h>


int main()

{

        int new_char;
        char t;

        int ctr=0;

        char str[100];

        printf("\n Replace the spaces of a string with a
specific character :\n");

    printf("--------------------------------------------------------
\n");

    printf(" Input a string : ");

        fgets(str, sizeof str, stdin);

    printf(" Input replace character : ");

        scanf("%c",&t);

        printf(" After replacing the space with  %c the new
string is :\n",t);

        while (str[ctr])

        {

                new_char=str[ctr];

                if (isspace(new_char))

                new_char=t;
                putchar (new_char);

                ctr++;

        }

        printf("\n\n");

        return 0;

}
```

output



```
 Replace the spaces of a string with a specific character :
------------------------------------------------------------
 Input a string : Greninja
 Input replace character : n
 After replacing the space with  n the new string is :
Greninjan
```

26. Write a program in C to count the number of punctuation characters exists in a string.

Test Data :

Input a string : The quick brown fox, jumps over the, lazy dog.

Expected Output :

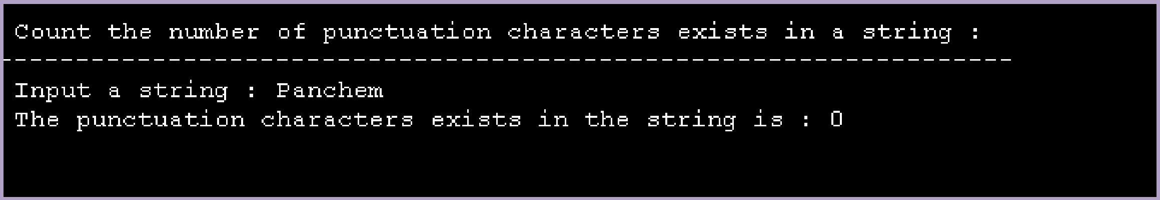
```
The punctuation characters exists in the string is : 3
```

Source code:

```
#include<stdio.h>
#include<ctype.h>

int main()
{
	int ctr1=0;
	int ctr2=0;
	char str[100];
	printf("\n Count the number of punctuation 
characters exists in a string :\n");
    printf("-------------------------------------------------------------
---\n");
    printf(" Input a string : ");
	fgets(str, sizeof str, stdin);
```

```
        while (str[ctr1])
        {
                if (ispunct(str[ctr1])) ctr2++;
                ctr1++;
        }
        printf (" The punctuation characters exists in the string is : %d\n\n", ctr2);
        return 0;
}
```

output:



```
Count the number of punctuation characters exists in a string :
------------------------------------------------------------------
Input a string : Panchem
The punctuation characters exists in the string is : 0
```

27. Write a program in C to print only the string before new line character.

Note: isprint() will only print line one, because the newline character is not printable.

Expected Output :

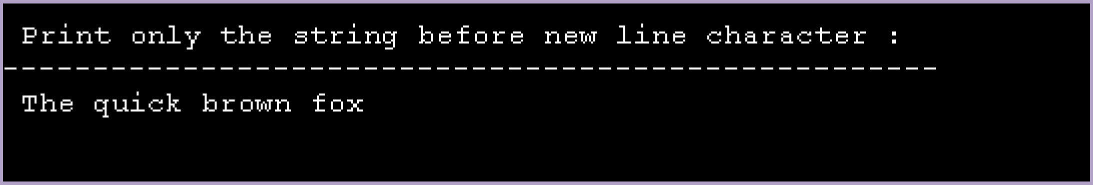
The quick brown fox

Source code:

```
#include<stdio.h>

#include<ctype.h>


int main()

{

        int ctr=0;

        char str[]=" The quick brown fox \n jumps over the
\n lazy dog. \n";

        printf("\n Print only the string before new line
character :\n");

    printf("-----------------------------------------------\n");

        while (isprint(str[ctr]))

        {

                putchar (str[ctr]);
```

```
            ctr++;
        }
        printf("\n\n");
        return 0;
}
```

output:



```
Print only the string before new line character :
--------------------------------------------------
The quick brown fox
```

28. Write a program in C to check whether a letter is lowercase or not.

Test Data :

Input a character : w

Expected Output :

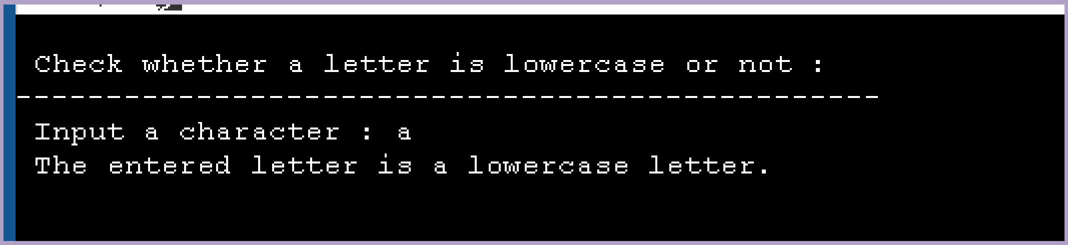
The entered letter is a lowercase letter.

Source code:

```
#include<stdio.h>

#include<ctype.h>


int main()

{

        char TestChar;

        printf("\n Check whether a letter is lowercase or not
:\n");

    printf("------------------------------------------------\n");

    printf(" Input a character : ");

    scanf( "%c", &TestChar );

        if( islower(TestChar) )

                printf( " The entered letter is a lowercase letter.
\n" );

        else

                printf( " The entered letter is not a lowercase
letter. \n" );
```

```
    return 0;
}
```

output:



```
 Check whether a letter is lowercase or not :
-------------------------------------------------
 Input a character : a
 The entered letter is a lowercase letter.
```

29. Write a program in C to read a file and remove the spaces between two words of its content.

Expected Output :

The content of the file is :

The quick brown fox jumps over the lazy dog

After removing the spaces the content is :

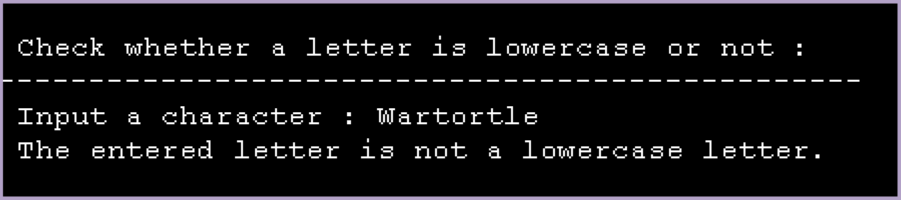
Thequickbrownfoxjumpsoverthelazydog

Source code:

```
#include<stdio.h>
```

```
#include<ctype.h>


int main()

{

        char TestChar;

        printf("\n Check whether a letter is lowercase or not
:\n");

    printf("----------------------------------------------\n");

    printf(" Input a character : ");

    scanf( "%c", &TestChar );

        if( islower(TestChar) )

                printf( " The entered letter is a lowercase letter.
\n" );

        else

                printf( " The entered letter is not a lowercase
letter. \n" );

        return 0;

}
```

output:



```
Check whether a letter is lowercase or not :
----------------------------------------------------
Input a character : Wartortle
The entered letter is not a lowercase letter.
```

30. Write a program in C to check whether a character is digit or not.

Test Data :

Input a character : 8

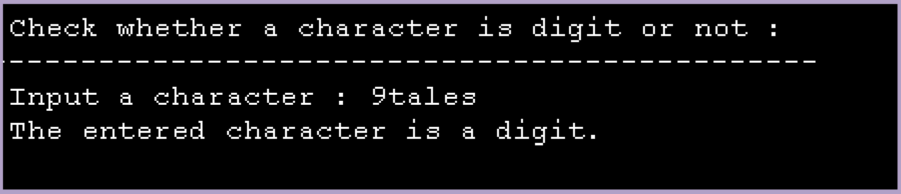
Expected Output :

The entered character is a digit.

Source code:

```
#include<stdio.h>

#include<ctype.h>


int main()

{
        char TestChar;

        printf("\n Check whether a character is digit or not
:\n");

    printf("-----------------------------------------\n");

    printf(" Input a character : ");

    scanf( "%c", &TestChar );

        if( isdigit(TestChar) )

            printf( " The entered character is a digit. \n\n" );

        else

            printf( " The entered character is not a digit.
\n\n" );

        return 0;

}
```

output:



```
Check whether a character is digit or not :
-------------------------------------------
Input a character : 9tales
The entered character is a digit.
```

31. Write a program in C to split string by space into words.

Test Data :

Input a string : this is a test string

Expected Output :

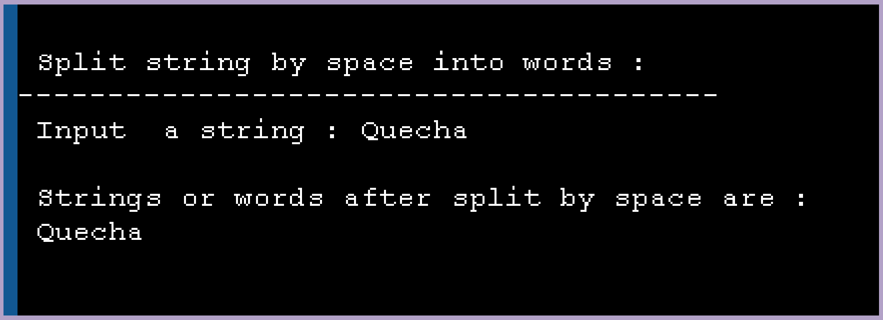
```
Strings or words after split by space are :


this

is

a

test

string .
```

Source code:

```
#include <stdio.h>

#include <string.h>

int main()
```

```
{
    char str1[100];
    char newString[10][10];
    int i,j,ctr;
       printf("\n\n Split string by space into words :\n");
       printf("---------------------------------------\n");

    printf(" Input  a string : ");
    fgets(str1, sizeof str1, stdin);

    j=0; ctr=0;
    for(i=0;i<=(strlen(str1));i++)
    {
        // if space or NULL found, assign NULL into
newString[ctr]
        if(str1[i]==' '||str1[i]=='\0')
        {
            newString[ctr][j]='\0';
            ctr++;  //for next word
            j=0;    //for next word, init index to 0
        }
        else
        {
            newString[ctr][j]=str1[i];
            j++;
        }
    }
    printf("\n Strings or words after split by space are
:\n");
    for(i=0;i < ctr;i++)
        printf(" %s\n",newString[i]);
    return 0;
}
```

output:



```
Split string by space into words :
---------------------------------------
Input  a string : Quecha

Strings or words after split by space are :
Quecha
```

32. Write a C programming to find the repeated character in a given string.

Test Data :

Input a string: PDEU

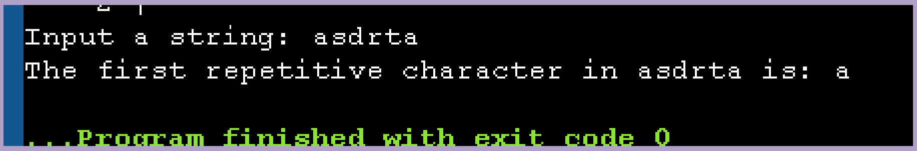
Expected Output:

Input a string: The first repetitive character in PDEU is: r

Source code:

```
#include<stdio.h>

#include

int ifexists(char p, char q[],  int v)

{

        int i;

        for (i=0; i<v;i++)
            if (q[i]==p) return (1);
        return (0);
}
int main()
{
        char string1[80],string2[80];
        int n,i,x;
        printf("Input a string: ");
        scanf("%s",string1);
        n=strlen(string1);
        string2[0]=string1[0];
        x=1;
        for(i=1;i < n;  i++)
        {
            if(ifexists(string1[i], string2, x))
            {
                printf("The first repetitive character in %s
is: %c ", string1, string1[i]);
                break;
            }
            else
            {
                string2[x]=string1[i];
                x++;
            }
        }
        if(i==n)
            printf("There is no repetitve character in the
string %s.", string1);
}
```

output:



```
Input a string: asdrta
The first repetitive character in asdrta is: a

...Program finished with exit code 0
```

33. Write a C program to count of each character in a given string.

Test Data :

Input a string: PDEU

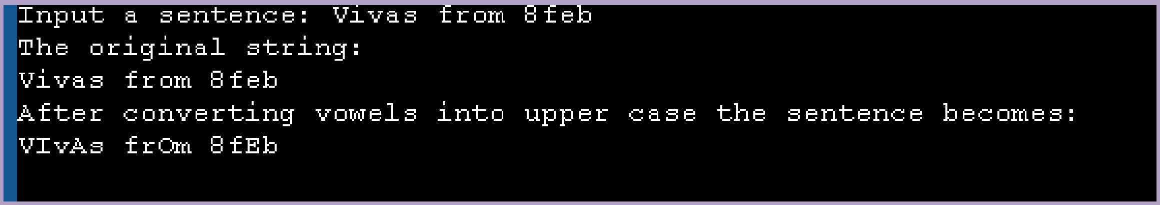
Expected Output:

```
Enter a str1ing: The count of each character in the string
PDEU is

w    1

3    1

r    2

e    2

s    1

o    1

u    1

c    1
```

Source code:

```
#include <stdio.h>

int main()

{

    char string1[255];

    int  i;

   printf("Input a sentence: ");

   gets(string1);

   printf("The original string:\n");

   puts(string1);

   i=0;

    while(string1[i]!='\0')

    {

        if(string1[i]=='a' ||string1[i]=='e' ||string1[i]=='i'
||string1[i]=='o' ||string1[i]=='u')

            string1[i]=string1[i]-32;
        i++;
    }
    printf("After converting vowels into upper case the
sentence becomes:\n");
    puts(string1);
}
```

output:



```
Input a sentence: Vivas from 8feb
The original string:
Vivas from 8feb
After converting vowels into upper case the sentence becomes:
VIvAs frOm 8fEb
```

34. Write a C programming to convert vowels into upper case character in a given string.

Test Data :

Input a string : PDEU

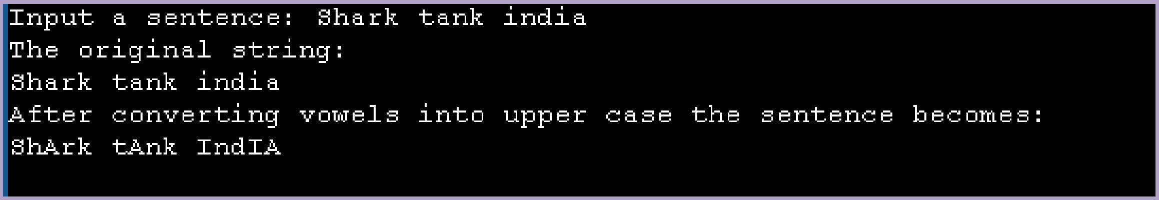
Expected Output:

```
Input a sentence: The original string:

PDEU

After converting vowels into upper case the sentence
becomes:

PDEU
```

Source code:

```c
#include <stdio.h>


int main()

{

    char string1[255];

    int  i;

   printf("Input a sentence: ");

   gets(string1);

   printf("The original string:\n");

   puts(string1);
   i=0;
    while(string1[i]!='\0')
    {
       if(string1[i]=='a' ||string1[i]=='e' ||string1[i]=='i'
||string1[i]=='o' ||string1[i]=='u')
          string1[i]=string1[i]-32;
       i++;
    }
    printf("After converting vowels into upper case the
sentence becomes:\n");
    puts(string1);
}
```

output:



```
Input a sentence: Shark tank india
The original string:
Shark tank india
After converting vowels into upper case the sentence becomes:
ShArk tAnk IndIA
```

FILE HANDLING

1.) Write a program in C to create and store information in a text file.
Test Data :

Input a sentence for the file : This is the content of the file test.txt.
Expected Output :

```
 The file test.txt created
successfully...!!
```

```
int main()

{

   char str[1000];

   FILE *fptr;

   char fname[20]="test.txt";


printf("\n\n Create a file (test.txt) and input text :\n");

        printf("----------------------------------------------\n");

fptr=fopen(fname,"w");

   if(fptr==NULL)

   {

printf(" Error in opening file!");

      exit(1);
   }
printf(" Input a sentence for the file : ");
fgets(str, sizeof str, stdin);
fprintf(fptr,"%s",str);
fclose(fptr);
printf("\n The file %s created
successfully...!!\n\n",fname);
   return 0;
}
```

2. Write a program in C to read an existing file.
Test Data :
Input the file name to be opened : test.txt
Expected Output :

```
 The content of the file test.txt is
:
```

```
This is the content of the file
test.txt.
```

```
void main()

{

        FILE *fptr;

        char fname[20];

        char str;

printf("\n\n Read an existing file :\n");

        printf("------------------------------\n");

        printf(" Input the filename to be opened : ");

        scanf("%s",fname);

        fptr = fopen (fname, "r");

        if (fptr == NULL)

        {

                printf(" File does not exist or cannot be
opened.\n");

                exit(0);
        }
        printf("\n The content of the file %s is  :\n",fname);
        str = fgetc(fptr);
        while (str != EOF)
            {
                printf ("%c", str);
                str = fgetc(fptr);
            }
        fclose(fptr);
    printf("\n\n");
```

3. Write a program in C to write multiple lines in a text file.
Test Data :
Input the number of lines to be written : 4
:: The lines are ::
test line 1
test line 2
test line 3

test line 4

Expected Output :

```
 The content of the file test.txt is
:
test line 1
test line 2
test line 3
test line 4
```

```
int main ()
{
  FILE * fptr;
  int i,n;
  char str[100];
  char fname[20]="test.txt";
  char str1;

printf("\n\n Write multiple lines in a text file and read
the file :\n");
```

```
        printf("------------------------------------------------------------
\n");

        printf(" Input the number of lines to be written : ");

        scanf("%d", &n);

        printf("\n :: The lines are ::\n");

        fptr = fopen (fname,"w");

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

                {

                fgets(str, sizeof str, stdin);

                fputs(str, fptr);

                }

fclose (fptr);

/*------------- read the file -------------------------------------*/

        fptr = fopen (fname, "r");

        printf("\n The content of the file %s is  :\n",fname);

        str1 = fgetc(fptr);

        while (str1 != EOF)
        {
              printf ("%c", str1);
              str1 = fgetc(fptr);
        }
printf("\n\n");
fclose (fptr);
  return 0;
}
```

4. Write a program in C to read the file and store the lines into an array.
Test Data :
Input the file name to be opened : test.txt
Expected Output :

```
 The content of the file test.txt  are
:
 test line 1
 test line 2
 test line 3
 test line 4
```

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>



#define LSIZ 128

#define RSIZ 10



int main(void)

{

   char line[RSIZ][LSIZ];
```

```
        char fname[20];
    FILE *fptr = NULL;
    int i = 0;
    int tot = 0;
printf("\n\n Read the file and store the lines into an array
:\n");
        printf("------------------------------------------------------\n");
        printf(" Input the filename to be opened : ");
        scanf("%s",fname);

fptr = fopen(fname, "r");
    while(fgets(line[i], LSIZ, fptr))
        {
        line[i][strlen(line[i]) - 1] = '\0';
i++;
    }
    tot = i;
```

```
        printf("\n The content of the file %s  are :
\n",fname);
    for(i = 0; i< tot; ++i)
    {
printf(" %s\n", line[i]);
    }
printf("\n");
    return 0;
}
```

5. Write a program in C to Find the Number of Lines in a Text File.
Test Data :
Input the file name to be opened : test.txt
Expected Output :

```
 The lines in the file test.txt are : 
4
```

```
#include <stdio.h>



#define FSIZE 100



int main()

{

   FILE *fptr;

   int ctr = 0;

   char fname[FSIZE];

   char c;
```

```
printf("\n\n Read the file and count the number of lines
:\n");
        printf("-----------------------------------------------\n");
        printf(" Input the file name to be opened : ");
        scanf("%s",fname);


fptr = fopen(fname, "r");
    if (fptr == NULL)
    {
printf("Could not open file %s", fname);
        return 0;
    }
    // Extract characters from file and store in character c
    for (c = getc(fptr); c != EOF; c = getc(fptr))
        if (c == '\n') // Increment count if this character is
newline
ctr = ctr + 1;
```

```
fclose(fptr);

printf(" The lines in the file %s are : %d \n \n", fname, ctr-
1);

    return 0;

}
```

6. Write a program in C to find the content of the file and number of lines in a Text File.
Test Data :
Input the filename to be opened : test.txt
Expected Output :

```
 The content of the file test.txt  are
:
 test line 1
 test line 2
 test line 3
 test line 4
 The lines in the file are : 4
```

```
#include <stdio.h>



#define FSIZE 100



int main()

{

   FILE *fptr;
```

```
    int ctr = 0;

    char fname[FSIZE];

    char c;

printf("\n\n Read the file and count the number of lines
:\n");

        printf("-----------------------------------------------\n");

        printf(" Input the file name to be opened : ");

        scanf("%s",fname);


fptr = fopen(fname, "r");

    if (fptr == NULL)

    {

printf("Could not open file %s", fname);

        return 0;

    }

    // Extract characters from file and store in character c

    for (c = getc(fptr); c != EOF; c = getc(fptr))
```

```
        if (c == '\n') // Increment count if this character is
newline

ctr = ctr + 1;

fclose(fptr);

printf(" The lines in the file %s are : %d \n \n", fname, ctr-
1);

    return 0;

}
```

7. Write a program in C to count a number of words and characters in a file.
Test Data :
Input the file name to be opened : test.txt
Expected Output :

```
 The content of the file test.txt are
:
test line 1
test line 2
test line 3
test line 4
 The number of words in the  file
test.txt are : 12
 The number of characters in the  file
test.txt are : 36
```

```
#include <stdio.h>

#include <stdlib.h>



void main()

{

   FILE *fptr;

   char ch;
```

```
    int wrd=1,charctr=1;

    char fname[20];

printf("\n\n Count the number of words and characters
in a file :\n");

        printf("------------------------------------------------------
\n");

        printf(" Input the filename to be opened : ");

        scanf("%s",fname);


fptr=fopen(fname,"r");

    if(fptr==NULL)

     {

printf(" File does not exist or can not be opened.");

      }

    else

        {

ch=fgetc(fptr);
```

```
printf(" The content of the file %s are : ",fname);
          while(ch!=EOF)
            {
printf("%c",ch);
                if(ch==' '||ch=='\n')
                    {
wrd++;
                    }
                    else
                    {
charctr++;
                    }
ch=fgetc(fptr);
            }
printf("\n The number of words in the  file %s are :
%d\n",fname,wrd-2);
```

```
printf(" The number of characters in the  file %s are :
%d\n\n",fname,charctr-1);

        }

fclose(fptr);

}
```

8. Write a program in C to delete a specific line from a file.

```
Assume that the content of the file
test.txt is :
test line 1
test line 2
test line 3
test line 4
```

Test Data :
Input the file name to be opened : test.txt
Input the line you want to remove : 2
Expected Output :

```
 The content of the file test.txt is :
test line 1
test line 3
test line 4
```

```
#include <stdio.h>

#include <string.h>



#define MAX 256



  int main()
```

```
 {
        int lno, ctr = 0;
        char ch;
        FILE *fptr1, *fptr2;
                char fname[MAX];
        char str[MAX], temp[] = "temp.txt";
                printf("\n\n Delete a specific line from a file
:\n");
                printf("-----------------------------------------\n");
                printf(" Input the file name to be opened : ");
                scanf("%s",fname);
        fptr1 = fopen(fname, "r");
        if (!fptr1)
                {
printf(" File not found or unable to open the input
file!!\n");
                return 0;
        }
        fptr2 = fopen(temp, "w"); // open the temporary file
in write mode
        if (!fptr2)
                {
printf("Unable to open a temporary file to write!!\n");
fclose(fptr1);
                return 0;
        }
printf(" Input the line you want to remove : ");
scanf("%d", &lno);
                lno++;
        // copy all contents to the temporary file except the
specific line
        while (!feof(fptr1))
        {
strcpy(str, "\0");
```

```
fgets(str, MAX, fptr1);
            if (!feof(fptr1))
            {
ctr++;
                /* skip the line at given line number */
                if (ctr != lno)
                {
fprintf(fptr2, "%s", str);
                }
            }
        }
fclose(fptr1);
fclose(fptr2);
        remove(fname);          // remove the original file
        rename(temp, fname);       // rename the temporary
file to original name
/*------ Read the file ----------------*/
```

```
   fptr1=fopen(fname,"r");
ch=fgetc(fptr1);
printf(" Now the content of the file %s is : \n",fname);
          while(ch!=EOF)
            {
printf("%c",ch);
ch=fgetc(fptr1);
            }
fclose(fptr1);
/*------- End of reading ---------------*/
        return 0;


  }
```

9. Write a program in C to replace a specific line with another text in a file.

```
Assume that the content of the file
test.txt is :
test line 1
test line 2
test line 3
test line 4
```

Test Data :
Input the file name to be opened : test.txt
Input the content of the new line : Yes, I am the new text instead of test line 2
Input the line no you want to replace : 2
Expected Output :

```
Replacement did successfully..!!
```

```
#include <stdio.h>

#include <string.h>



#define MAX 256



  int main()
```

```
 {
        FILE *fptr1, *fptr2;
        int lno, linectr = 0;
        char str[MAX],fname[MAX];
        char newln[MAX], temp[] = "temp.txt";

                printf("\n\n Replace a specific line in a text file
with a new text :\n");
                printf("-----------------------------------------------------
-------\n");
                printf(" Input the file name to be opened : ");
fgets(fname, MAX, stdin);
fname[strlen(fname) - 1] = '\0';
        fptr1 = fopen(fname, "r");
        if (!fptr1)
        {
printf("Unable to open the input file!!\n");
```

```
                return 0;
        }
        fptr2 = fopen(temp, "w");
        if (!fptr2)
        {
printf("Unable to open a temporary file to write!!\n");
fclose(fptr1);
                return 0;
        }
        /* get the new line from the user */
printf(" Input the content of the new line : ");
fgets(newln, MAX, stdin);
        /* get the line number to delete the specific line */
printf(" Input the line no you want to replace : ");
scanf("%d", &lno);
lno++;
```

```
         // copy all contents to the temporary file other
except specific line
        while (!feof(fptr1))
        {
strcpy(str, "\0");
fgets(str, MAX, fptr1);
            if (!feof(fptr1))
            {
linectr++;
                if (linectr != lno)
                    {
fprintf(fptr2, "%s", str);
                    }
                    else
                    {
fprintf(fptr2, "%s", newln);
                    }
            }
        }
fclose(fptr1);
fclose(fptr2);
        remove(fname);
        rename(temp, fname);
printf(" Replacement did successfully..!! \n");
        return 0;
  }
```

10. Write a program in C to append multiple lines at the end of a text file.

```
Assume that the content of the file
test.txt is :
test line 1
test line 2
test line 3
test line 4
```

Test Data :
Input the file name to be opened : test.txt
Input the number of lines to be written : 3
The lines are :
test line 5
test line 6
test line 7
Expected Output :

```
The content of the file test.txt is  :
test line 1
test line 2
test line 3
test line 4

test line 5
test line 6
test line 7
```

```
#include <stdio.h>


int main ()

{

  FILE * fptr;
  int i,n;

  char str[100];

  char fname[20];

  char str1;


       printf("\n\n Append multiple lines at the end of a
text file :\n");

       printf("------------------------------------------------------\n");

       printf(" Input the file name to be opened : ");

       scanf("%s",fname);

fptr = fopen(fname, "a");

printf(" Input the number of lines to be written : ");

scanf("%d", &n);

printf(" The lines are : \n");

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

   {

fgets(str, sizeof str, stdin);
fputs(str, fptr);
  }
fclose (fptr);
//----- Read the file after appended -------
        fptr = fopen (fname, "r");
        printf("\n The content of the file %s is  :\n",fname);
        str1 = fgetc(fptr);
        while (str1 != EOF)
                {
                        printf ("%c", str1);
                        str1 = fgetc(fptr);
                }
printf("\n\n");
fclose (fptr);
//------- End of reading ------------------
  return 0;
}
```

11. Write a program in C to copy a file in another name.

```
Assume that the content of the file
test.txt is :
test line 1
test line 2
test line 3
test line 4
```

Test Data :
Input the source file name : test.txt
Input the new file name : test1.txt
Expected Output :

```
 The file test.txt  copied
successfully in the file test1.txt.
```

If you read the new file you will see the content of the file :

```
test line 1
test line 2
test line 3
test line 4
```

```
#include <stdio.h>

#include <stdlib.h>


void main()

{

        FILE *fptr1, *fptr2;

        char ch, fname1[20], fname2[20];


        printf("\n\n Copy a file in another name :\n");

        printf("----------------------------------\n");


        printf(" Input the source file name : ");

        scanf("%s",fname1);


        fptr1=fopen(fname1, "r");

        if(fptr1==NULL)

        {
            printf(" File does not found or error in
    opening.!!");
            exit(1);
        }
        printf(" Input the new file name : ");
        scanf("%s",fname2);
        fptr2=fopen(fname2, "w");
        if(fptr2==NULL)
        {
            printf(" File does not found or error in
    opening.!!");
            fclose(fptr1);
            exit(2);
        }
        while(1)
        {
            ch=fgetc(fptr1);
            if(ch==EOF)
            {
                break;
            }
            else
            {
                fputc(ch, fptr2);
            }
        }
        printf(" The file %s  copied successfully in the file %s.
\n\n",fname1,fname2);
        fclose(fptr1);
        fclose(fptr2);
        getchar();
}
```

12. Write a program in C to merge two files and write it in a new file.

```
Assume that the content of the file
test.txt and test1.txr is :
 The content of the file test.txt is
:
This is the file test.txt.

 The content of the file test1.txt is
:
This is the file test1.txt.
```

Test Data :
Input the 1st file name : test.txt
Input the 2nd file name : test1.txt
Input the new file name where to merge the above two files : mergefiles.txt
Expected Output :

```
 The two files merged into
mergefiles.txt file successfully..!!
```

Here is the content of the merge file mergefiles.txt :

```
 The content of the file
mergefiles.txt is  :
This is the file test.txt.
This is the file test1.txt.
```

```
#include <stdio.h>

#include <stdlib.h>
```

```
void main()
{
        FILE *fold1, *fold2, *fnew;
        char ch, fname1[20], fname2[20], fname3[30];

        printf("\n\n Merge two files and write it in a new file
:\n");
        printf("----------------------------------------------------\n");

        printf(" Input the 1st file name : ");
        scanf("%s",fname1);
        printf(" Input the 2nd file name : ");
        scanf("%s",fname2);
        printf(" Input the new file name where to merge the
above two files : ");
        scanf("%s",fname3);
        fold1=fopen(fname1, "r");
        fold2=fopen(fname2, "r");
        if(fold1==NULL || fold2==NULL)
        {
//              perror("Error Message ");
                printf(" File does not exist or error in
opening...!!\n");
                exit(EXIT_FAILURE);
        }
        fnew=fopen(fname3, "w");
        if(fnew==NULL)
        {
//              perror("Error Message ");
                printf(" File does not exist or error in
opening...!!\n");
                exit(EXIT_FAILURE);
        }
        while((ch=fgetc(fold1))!=EOF)
```

```
	{
		fputc(ch, fnew);
	}
	while((ch=fgetc(fold2))!=EOF)
	{
		fputc(ch, fnew);
	}
	printf(" The two files merged into %s file
successfully..!!\n\n", fname3);
	fclose(fold1);
	fclose(fold2);
	fclose(fnew);
}
```

13. Write a program in C to encrypt a text file.

```
 Assume that, the content of the file
test.txt is  :
Welcome to w3resource.com.
```

Test Data :
Input the name of file to encrypt : test.txt
Expected Output :

```
File test.txt successfully encrypted
..!!
```

If you read the file test.txt you will see the following :

```
������Ʉ�ҕ��������ɒ��ђn
```

```
#include <stdio.h>

#include <stdlib.h>


void main()

{

        char fname[20], ch;

        FILE *fpts, *fptt;


        printf("\n\n Encrypt a text file :\n");

        printf("--------------------------\n");


        printf(" Input the name of file to encrypt : ");
        scanf("%s",fname);


        fpts=fopen(fname, "r");
        if(fpts==NULL)
        {
                printf(" File does not exists or error in
opening..!!");
                exit(1);
        }
        fptt=fopen("temp.txt", "w");
        if(fptt==NULL)
        {
                printf(" Error in creation of file temp.txt ..!!");
                fclose(fpts);
                exit(2);
        }
        while(1)
```

```
{
        ch=fgetc(fpts);
        if(ch==EOF)
        {
                break;
        }
        else
        {
                ch=ch+100;
                fputc(ch, fptt);
        }
}
fclose(fpts);
fclose(fptt);
fpts=fopen(fname, "w");
if(fpts==NULL)
{
```

```
                printf(" File does not exists or error in
opening..!!");
                exit(3);
        }
        fptt=fopen("temp.txt", "r");
        if(fptt==NULL)
        {
                printf(" File does not exists or error in
opening..!!");
                fclose(fpts);
                exit(4);
        }
        while(1)
        {
                ch=fgetc(fptt);
                if(ch==EOF)
                {
```

```
                break;
            }
            else
            {
                fputc(ch, fpts);
            }
        }
        printf(" File %s successfully encrypted ..!!\n\n", 
fname);
        fclose(fpts);
        fclose(fptt);
}
```

14. Write a program in C to decrypt a previously encrypted file file.

```
 Assume that, the content of the file
test.txt was  :
������Ʉ�ʒﷱ�������ɒ��ђn
```

```
After encryption, the content of the
file is :
Welcome to w3resource.com.
```
Test Data :
Input the name of file to decrypt : test.txt
Expected Output :

```
The file test.txt decrypted
successfully..!!
```
Now, if you read the file test.txt you will see the following :

```
Welcome to w3resource.com.
```

```
#include <stdio.h>

#include <stdlib.h>



void main()

{

    char ch, fname[20];

    FILE *fpts, *fptt;



    printf("\n\n Decrypt a text file :\n");
```

```
        printf("--------------------------\n");


        printf(" Input the name of file to decrypt : ");
        scanf("%s",fname);


        fpts=fopen(fname, "w");
        if(fpts==NULL)
        {
                printf(" File does not exists or error in
opening..!!");
                exit(7);
        }
        fptt=fopen("temp.txt", "r");
        if(fptt==NULL)
        {
                printf(" File does not exists or error in
opening..!!");
```

```
        fclose(fpts);
        exit(9);
    }
    while(1)
    {
        ch=fgetc(fptt);
        if(ch==EOF)
        {
            break;
        }
        else
        {
            ch=ch-100;
            fputc(ch, fpts);
        }
    }
```

```
        printf(" The file %s decrypted
successfully..!!\n\n",fname);

        fclose(fpts);

        fclose(fptt);

}
```

15. Write a program in C to remove a file from the disk.
Test Data :
Input the name of file to delete : test.txt
Expected Output :

```
 The file test.txt is deleted
successfully..!!!
```

```
#include <stdio.h>


void main()

{

	int status;

	char fname[20];

	printf("\n\n Remove a file from the disk :\n");

	printf("--------------------------------\n");

	printf(" Input the name of file to delete : ");

	scanf("%s",fname);

	status=remove(fname);
```

```
        if(status==0)
        {
                printf(" The file %s is deleted 
successfully..!!\n\n",fname);
        }
        else
        {
                printf(" Unable to delete file %s\n\n",fname);
        }
}
```

16)Write a C program to create a file and write contents, save and close the file.

```
#include <stdio.h>
#include <stdlib.h>


#define DATA_SIZE 1000
```

```
int main()
{
    /* Variable to store user content */
    char data[DATA_SIZE];

    /* File pointer to hold reference to our file */
    FILE * fPtr;



    /*
     * Open file in w (write) mode.
     * "data/file1.txt" is complete path to create file
     */
fPtr = fopen("data/file1.txt", "w");
```

```
    /* fopen() return NULL if last operation was
unsuccessful */
    if(fPtr == NULL)
    {
        /* File not created hence exit */
printf("Unable to create file.\n");
        exit(EXIT_FAILURE);
    }



    /* Input contents from user to store in file */
printf("Enter contents to store in file : \n");
fgets(data, DATA_SIZE, stdin);



    /* Write data to file */
fputs(data, fPtr);
```

```
    /* Close file to save file data */
fclose(fPtr);



    /* Success message */
printf("File created and saved successfully. :) \n");



    return 0;
}
```

17.) Write a C program to read file contents and display on console.

```
#include <stdio.h>
```

```
#include <stdlib.h>



int main()

{

    /* File pointer to hold reference to our file */

    FILE * fPtr;


    char ch;




    /*

     * Open file in r (read) mode.

     * "data/file1.txt" is complete file path to read

     */

fPtr = fopen("data/file1.txt", "r");
```

```
    /* fopen() return NULL if last operation was
unsuccessful */

    if(fPtr == NULL)

    {

        /* Unable to open file hence exit */

printf("Unable to open file.\n");

printf("Please check whether file exists and you have
read privilege.\n");

        exit(EXIT_FAILURE);

    }




    /* File open success message */

printf("File opened successfully. Reading file contents
character by character. \n\n");



    do
    {
        /* Read single character from file */
ch = fgetc(fPtr);

        /* Print character read on console */
putchar(ch);

    } while(ch != EOF); /* Repeat this if last read character
is not EOF */



    /* Done with this file, close file to release resource */
fclose(fPtr);



    return 0;
}
```

18)Write a C program to read numbers from a file and write even, odd and prime numbers to separate file.

```
#include <stdio.h>

#include <stdlib.h>



/* Function declarations */

int isEven(const int NUM);

int isPrime(const int NUM);



int main()

{

    /* File pointer to hold reference to different files */

    FILE * fPtrIn,
        * fPtrEven,
        * fPtrOdd,
        * fPtrPrime;



    int num, success;


    /*
     * Open all files to perform read/write.
     */
fPtrIn   = fopen("data/numbers.txt", "r");
fPtrEven = fopen("data/even-numbers.txt" , "w");
fPtrOdd  = fopen("data/odd-numbers.txt"  , "w");
fPtrPrime= fopen("data/prime-numbers.txt", "w");
```

```
    /* fopen() return NULL if unable to open file in given
mode. */
    if(fPtrIn == NULL || fPtrEven == NULL || fPtrOdd ==
NULL || fPtrPrime == NULL)
    {
        /* Unable to open file hence exit */
printf("Unable to open file.\n");
printf("Please check whether file exists and you have
read/write privilege.\n");
        exit(EXIT_FAILURE);
    }
```

19)Write a C program to append content to a file.

```
#include <stdio.h>
#include <stdlib.h>
```

```
#define BUFFER_SIZE 1000



void readFile(FILE * fPtr);



int main()
{
    /* File pointer to hold reference of input file */
    FILE *fPtr;
    char filePath[100];

    char dataToAppend[BUFFER_SIZE];



    /* Input file path to remove empty lines from user */
printf("Enter file path: ");
```

```
scanf("%s", filePath);


    /*  Open all file in append mode. */
fPtr = fopen(filePath, "a");




    /* fopen() return NULL if unable to open file in given
mode. */
    if (fPtr == NULL)
    {
        /* Unable to open file hence exit */
printf("\nUnable to open '%s' file.\n", filePath);
printf("Please check whether file exists and you have
write privilege.\n");
        exit(EXIT_FAILURE);
    }
    /* Input data to append from user */
printf("\nEnter data to append: ");
fflush(stdin);          // To clear extra white space
characters in stdin
fgets(dataToAppend, BUFFER_SIZE, stdin);



    /* Append data to file */
fputs(dataToAppend, fPtr);



    /* Reopen file in read mode to print file contents */
fPtr = freopen(filePath, "r", fPtr);

    /* Print file contents after appending string */
printf("\nSuccessfully appended data to file. \n");
printf("Changed file contents:\n\n");
readFile(fPtr);



    /* Done with file, hence close file. */
fclose(fPtr);


    return 0;
}




/**
 * Reads a file character by character
 * and prints on console.
 *
 * @fPtr    Pointer to FILE to read.
 */
void readFile(FILE * fPtr)
{
    char ch;

    do
    {
ch = fgetc(fPtr);

putchar(ch);

    } while (ch != EOF);
}
```

20)Write a C program to compare two files.

```
#include <stdio.h>
```

```
#include <stdlib.h>


/* Function declaration */
int compareFile(FILE * fPtr1, FILE * fPtr2, int * line, int *
col);



int main()
{
    /* File pointer to hold reference of input file */
    FILE * fPtr1;
    FILE * fPtr2;
    char path1[100];
    char path2[100];

    int diff;
    int line, col;
    /* Input path of files to compare */
printf("Enter path of first file: ");
scanf("%s", path1);
printf("Enter path of second file: ");
scanf("%s", path2);



    /*  Open all files to compare */
    fPtr1 = fopen(path1, "r");
    fPtr2 = fopen(path2, "r");

    /* fopen() return NULL if unable to open file in given
mode. */
    if (fPtr1 == NULL || fPtr2 == NULL)
    {
        /* Unable to open file hence exit */
    printf("\nUnable to open file.\n");
    printf("Please check whether file exists and you have
    read privilege.\n");
        exit(EXIT_FAILURE);
    }



    /* Call function to compare file */
    diff = compareFile(fPtr1, fPtr2, &line, &col);


    if (diff == 0)
    {
printf("\nBoth files are equal.");
    }
    else
    {
```

```
printf("\nFiles are not equal.\n");
printf("Line: %d, col: %d\n", line, col);
    }


    /* Finally close files to release resources */
fclose(fPtr1);
fclose(fPtr2);

    return 0;
}


/**
 * Function to compare two files.
 * Returns 0 if both files are equivalent, otherwise
returns
```

```
 * -1 and sets line and col where both file differ.
 */
int compareFile(FILE * fPtr1, FILE * fPtr2, int * line, int *
col)
{
    char ch1, ch2;


    *line = 1;
    *col  = 0;


    do
    {
        // Input character from both files
        ch1 = fgetc(fPtr1);
        ch2 = fgetc(fPtr2);


        // Increment line
        if (ch1 == '\n')
        {
            *line += 1;
            *col = 0;
        }

        // If characters are not same then return -1
        if (ch1 != ch2)
            return -1;

        *col  += 1;

    } while (ch1 != EOF && ch2 != EOF);



    /* If both files have reached end */
    if (ch1 == EOF && ch2 == EOF)
```

```
        return 0;
    else
        return -1;
}
```

21)Write a C program to copy contents from one file to another file.

```
int main()
{
    FILE *sourceFile;
    FILE *destFile;
    char sourcePath[100];
    char destPath[100];

    char ch;

    /* Input path of files to copy */
```

```
printf("Enter source file path: ");
scanf("%s", sourcePath);
printf("Enter destination file path: ");
scanf("%s", destPath);

    /*
     * Open source file in 'r' and
     * destination file in 'w' mode
     */
sourceFile  = fopen(sourcePath, "r");
destFile    = fopen(destPath,   "w");

    /* fopen() return NULL if unable to open file in given
mode. */
    if (sourceFile == NULL || destFile == NULL)
    {
        /* Unable to open file hence exit */
```

```
printf("\nUnable to open file.\n");

printf("Please check if file exists and you have read/write
privilege.\n");


        exit(EXIT_FAILURE);

    }




    /*

     * Copy file contents character by character.

     */

ch = fgetc(sourceFile);

    while (ch != EOF)

    {

        /* Write to destination file */

fputc(ch, destFile);
```

```
        /* Read next character from source file */
ch = fgetc(sourceFile);
    }



printf("\nFiles copied successfully.\n");



    /* Finally close files to release resources */
fclose(sourceFile);
fclose(destFile);

    return 0;
}
```

22)Write a C program to merge two file to third file.

```
#include <stdio.h>
```

```
#include <stdlib.h>



int main()
{
    FILE *sourceFile1;
    FILE *sourceFile2;
    FILE *destFile;
    char sourcePath1[100];
    char sourcePath2[100];
    char destPath[100];

    char ch;

    /* Input path of files to merge to third file */
printf("Enter first source file path: ");
scanf("%s", sourcePath1);
```

```
printf("Enter second source file path: ");

scanf("%s", sourcePath2);

printf("Enter destination file path: ");

scanf("%s", destPath);


    /*

     * Open source files in 'r' and

     * destination file in 'w' mode

     */

    sourceFile1 = fopen(sourcePath1, "r");

    sourceFile2 = fopen(sourcePath2, "r");

destFile    = fopen(destPath,    "w");




    /* fopen() return NULL if unable to open file in given
mode. */
```

```
    if (sourceFile1 == NULL || sourceFile2 == NULL ||
destFile == NULL)
    {
        /* Unable to open file hence exit */
printf("\nUnable to open file.\n");
printf("Please check if file exists and you have read/write
privilege.\n");


        exit(EXIT_FAILURE);
    }




    /* Copy contents of first file to destination */
    while ((ch = fgetc(sourceFile1)) != EOF)
fputc(ch, destFile);


    /* Copy contents of second file to destination */
    while ((ch = fgetc(sourceFile2)) != EOF)
fputc(ch, destFile);

printf("\nFiles merged successfully to '%s'.\n", destPath);

    /* Close files to release resources */
fclose(sourceFile1);
fclose(sourceFile2);
fclose(destFile);

    return 0;
}
```

23)Write a C program to count characters, words and lines in a text file

```
int main()

{

    FILE * file;

    char path[100];


    char ch;

    int characters, words, lines;




    /* Input path of files to merge to third file */

printf("Enter source file path: ");

scanf("%s", path);


    /* Open source files in 'r' mode */

    file = fopen(path, "r");
    /* Check if file opened successfully */
    if (file == NULL)
    {
printf("\nUnable to open file.\n");
printf("Please check if file exists and you have read
privilege.\n");

        exit(EXIT_FAILURE);
    }

    /*
     * Logic to count characters, words and lines.
     */
    characters = words = lines = 0;
    while ((ch = fgetc(file)) != EOF)
    {
        characters++;
        /* Check new line */
        if (ch == '\n' || ch == '\0')
            lines++;

        /* Check words */
        if (ch == ' ' || ch == '\t' || ch == '\n' || ch == '\0')
            words++;
    }

    /* Increment words and lines for last word */
    if (characters > 0)
    {
        words++;
        lines++;
    }
```

```
    /* Print file statistics */
printf("\n");
printf("Total characters = %d\n", characters);
printf("Total words      = %d\n", words);
printf("Total lines      = %d\n", lines);



    /* Close files to release resources */
fclose(file);

    return 0;
}
```

24)Write a C program to remove a word from text file.

```
#include <stdio.h>
#include <stdlib.h>
```

```
#include <string.h>

#define BUFFER_SIZE 1000


void removeAll(char * str, const char * toRemove);


int main()
{
    FILE * fPtr;
    FILE * fTemp;
    char path[100];

    char toRemove[100];
    char buffer[1000];
    /* Input source file path path */
printf("Enter path of source file: ");
scanf("%s", path);

printf("Enter word to remove: ");
scanf("%s", toRemove);


    /*  Open files */
fPtr  = fopen(path, "r");
fTemp = fopen("delete.tmp", "w");

    /* fopen() return NULL if unable to open file in given
mode. */
    if (fPtr == NULL || fTemp == NULL)
    {
        /* Unable to open file hence exit */
printf("\nUnable to open file.\n");
printf("Please check whether file exists and you have
read/write privilege.\n");
        exit(EXIT_SUCCESS);
    }



    /*
     * Read line from source file and write to destination
     * file after removing given word.
     */
    while ((fgets(buffer, BUFFER_SIZE, fPtr)) != NULL)
    {
        // Remove all occurrence of word from current line
removeAll(buffer, toRemove);
```

```
        // Write to temp file
fputs(buffer, fTemp);
    }



    /* Close all files to release resource */
fclose(fPtr);
fclose(fTemp);



    /* Delete original source file */
    remove(path);


    /* Rename temp file as original file */
    rename("delete.tmp", path);
```

```
printf("\nAll occurrence of '%s' removed successfully.",
toRemove);


    return 0;
}




/**
 * Remove all occurrences of a given word in string.
 */
void removeAll(char * str, const char * toRemove)
{
    int i, j, stringLen, toRemoveLen;
    int found;

stringLen   = strlen(str);      // Length of string
```

```
toRemoveLen = strlen(toRemove); // Length of word to
remove



    for(i=0; i<= stringLen - toRemoveLen; i++)
    {
        /* Match word with string */
        found = 1;
        for(j=0; j <toRemoveLen; j++)
        {
            if(str[i + j] != toRemove[j])
            {
                found = 0;
                break;
            }
        }
        /* If it is not a word */
        if(str[i + j] != ' ' && str[i + j] != '\t' && str[i + j] != '\n'
    && str[i + j] != '\0')
        {
            found = 0;
        }

        /*
         * If word is found then shift all characters to left
         * and decrement the string length
         */
        if(found == 1)
        {
            for(j=i; j <= stringLen - toRemoveLen; j++)
            {
                str[j] = str[j + toRemoveLen];
            }
```

```
stringLen = stringLen - toRemoveLen;


            // We will match next occurrence of word from
current index.
i--;
        }
    }
}
```

25)Write a C program to remove specific line from a text file.

```
#include <stdio.h>
#include <stdlib.h>

#define BUFFER_SIZE 1000
```

```
/* Function declarations */
void deleteLine(FILE *srcFile, FILE *tempFile, const int
line);
void printFile(FILE *fptr);



int main()
{
    FILE *srcFile;
    FILE *tempFile;

    char path[100];

    int line;
    /* Input file path and line number */
printf("Enter file path: ");
scanf("%s", path);

printf("Enter line number to remove: ");
scanf("%d", &line);



    /* Try to open file */
srcFile  = fopen(path, "r");
tempFile = fopen("delete-line.tmp", "w");



    /* Exit if file not opened successfully */
    if (srcFile == NULL || tempFile == NULL)
    {
printf("Unable to open file.\n");
printf("Please check you have read/write previleges.\n");


        exit(EXIT_FAILURE);
    }




printf("\nFile contents before removing line.\n\n");
printFile(srcFile);



    // Move src file pointer to beginning
    rewind(srcFile);


    // Delete given line from file.
deleteLine(srcFile, tempFile, line);
```

```
    /* Close all open files */
fclose(srcFile);
fclose(tempFile);


    /* Delete src file and rename temp file as src */
    remove(path);
    rename("delete-line.tmp", path);


printf("\n\n\nFile contents after removing %d line.\n\n",
line);

    // Open source file and print its contents
srcFile = fopen(path, "r");
printFile(srcFile);
```

```
fclose(srcFile);


    return 0;
}
```

26)Write a C program to remove empty lines from a text file,

```
#include <stdio.h>
#include <stdlib.h>


#define BUFFER_SIZE 1000


/* Function declarations */
int  isEmpty(const char *str);
void removeEmptyLines(FILE *srcFile, FILE *tempFile);
void printFile(FILE *fptr);
```

```
int main()
{
    FILE *srcFile;
    FILE *tempFile;

    char path[100];



    /* Input file path */
printf("Enter file path: ");
scanf("%s", path);



    /* Try to open file */
srcFile  = fopen(path, "r");
tempFile = fopen("remove-blanks.tmp", "w");
```

```
    /* Exit if file not opened successfully */
    if (srcFile == NULL || tempFile == NULL)
    {
printf("Unable to open file.\n");
printf("Please check you have read/write previleges.\n");

        exit(EXIT_FAILURE);
    }




printf("\nFile contents before removing all empty
lines.\n\n");
printFile(srcFile);
```

```
    // Move src file pointer to beginning

    rewind(srcFile);


    // Remove empty lines from file.

removeEmptyLines(srcFile, tempFile);




    /* Close all open files */

fclose(srcFile);

fclose(tempFile);




    /* Delete src file and rename temp file as src */

    remove(path);

    rename("remove-blanks.tmp", path);
```

```
printf("\n\n\nFile contents after removing all empty
line.\n\n");

    // Open source file and print its contents
srcFile = fopen(path, "r");
printFile(srcFile);
fclose(srcFile);

    return 0;
}



/**
 * Print contents of a file.
 */
void printFile(FILE *fptr)
```

```
{
    char ch;


    while((ch = fgetc(fptr)) != EOF)
putchar(ch);
}




/**
 * Checks, whether a given string is empty or not.
 * A string is empty if it only contains white space
 * characters.
 *
 * Returns 1 if given string is empty otherwise 0.
 */
int isEmpty(const char *str)
```

```
{
    char ch;

    do
    {
ch = *(str++);

        // Check non whitespace character
        if(ch != ' ' &&ch != '\t' &&ch != '\n' &&ch != '\r' &&ch
!= '\0')
            return 0;

    } while (ch != '\0');

    return 1;
}
```

```
/**
 * Function to remove empty lines from a file.
 */
void removeEmptyLines(FILE *srcFile, FILE *tempFile)
{
    char buffer[BUFFER_SIZE];


    while ((fgets(buffer, BUFFER_SIZE, srcFile)) != NULL)
    {
        /* If current line is not empty then write to
temporary file */
        if(!isEmpty(buffer))
fputs(buffer, tempFile);
    }
}
```

27)Write a C program to find occurrence of a word in a text file.

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>



#define BUFFER_SIZE 1000



/* Function declarations */

int indexOf(FILE *fptr, const char *word, int *line, int
*col);



int main()
```

```
{
    FILE *fptr;
    char path[100];


    char word[50];


    int line, col;




    /* Input file path */
printf("Enter file path: ");
scanf("%s", path);


    /* Input word to search in file */
printf("Enter word to search in file: ");
scanf("%s", word);
```

```
    /* Try to open file */
fptr = fopen(path, "r");


    /* Exit if file not opened successfully */
    if (fptr == NULL)
    {
printf("Unable to open file.\n");
printf("Please check you have read/write previleges.\n");


        exit(EXIT_FAILURE);
    }




    // Find index of word in fptr
indexOf(fptr, word, &line, &col);
```

```
    if (line != -1)
printf("'%s' found at line: %d, col: %d\n", word, line + 1,
col + 1);
    else
printf("'%s' does not exists.", word);



    // Close file
fclose(fptr);

    return 0;
}



/**
 * Finds, first index of a word in given file. First index is
represented
```

```
 * using line and column.
 */
int indexOf(FILE *fptr, const char *word, int *line, int
*col)
{
    char str[BUFFER_SIZE];
    char *pos;

    *line = -1;
    *col  = -1;

    while ((fgets(str, BUFFER_SIZE, fptr)) != NULL)
    {
        *line += 1;

        // Find first occurrence of word in str
pos = strstr(str, word);
```

```
    if (pos != NULL)
    {
        // First index of word in str is
        // Memory address of pos - memory
        // address of str.
        *col = (pos - str);
        break;
    }
}


// If word is not found then set line to -1
if (*col == -1)
    *line = -1;

return *col;
```

```
}
```

28)Write a C program to count occurrences of a word in a text file.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define BUFFER_SIZE 1000

/* Function declarations */
int countOccurrences(FILE *fptr, const char *word);



int main()
{
   FILE *fptr;
```

```
    char path[100];

    char word[50];

    int wCount;

    /* Input file path */
printf("Enter file path: ");
scanf("%s", path);

    /* Input word to search in file */
printf("Enter word to search in file: ");
scanf("%s", word);

    /* Try to open file */
fptr = fopen(path, "r");
```

```
    /* Exit if file not opened successfully */
    if (fptr == NULL)
    {
printf("Unable to open file.\n");
printf("Please check you have read/write previleges.\n");

        exit(EXIT_FAILURE);
    }

    // Call function to count all occurrence of word
wCount = countOccurrences(fptr, word);

printf("'%s' is found %d times in file.", word, wCount);


    // Close file
fclose(fptr);
```

```
    return 0;
}



/**
 * Returns total occurrences of a word in given file.
 */
int countOccurrences(FILE *fptr, const char *word)
{
    char str[BUFFER_SIZE];
    char *pos;

    int index, count;

    count = 0;
```

```
// Read line from file till end of file.
while ((fgets(str, BUFFER_SIZE, fptr)) != NULL)
{
    index = 0;

    // Find next occurrence of word in str
    while ((pos = strstr(str + index, word)) != NULL)
    {
        // Index of word in str is
        // Memory address of pos - memory
        // address of str.
        index = (pos - str) + 1;

        count++;
    }
}
```

```
    return count;
}
```

29)Write a C program to count occurrences of all words in a text file.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

#define MAX_WORDS   1000




int main()
{
    FILE *fptr;
    char path[100];

    int i, len, index, isUnique;


    // List of distinct words

    char words[MAX_WORDS][50];

    char word[50];


    // Count of distinct words

    int  count[MAX_WORDS];




    /* Input file path */

printf("Enter file path: ");

scanf("%s", path);




    /* Try to open file */
```

```
fptr = fopen(path, "r");


    /* Exit if file not opened successfully */
    if (fptr == NULL)
    {
printf("Unable to open file.\n");
printf("Please check you have read previleges.\n");


        exit(EXIT_FAILURE);
    }


    // Initialize words count to 0
    for (i=0; i<MAX_WORDS; i++)
        count[i] = 0;
    index = 0;

    while (fscanf(fptr, "%s", word) != EOF)
    {
        // Convert word to lowercase
strlwr(word);

        // Remove last punctuation character
len = strlen(word);
        if (ispunct(word[len - 1]))
            word[len - 1] = '\0';



        // Check if word exits in list of all distinct words
isUnique = 1;
        for (i=0; i<index &&isUnique; i++)
        {
            if (strcmp(words[i], word) == 0)
isUnique = 0;
        }


        // If word is unique then add it to distinct words list
        // and increment index. Otherwise increment
occurrence
        // count of current word.
        if (isUnique)
        {
strcpy(words[index], word);
            count[index]++;


            index++;
        }
        else
        {
            count[i - 1]++;
        }
    }

    // Close file
fclose(fptr);



    /*
     * Print occurrences of all words in file.
     */
printf("\nOccurrences of all distinct words in file: \n");
    for (i=0; i<index; i++)
    {
        /*
         * %-15s prints string in 15 character width.
        * - is used to print string left align inside
        * 15 character width space.
        */
printf("%-15s => %d\n", words[i], count[i]);
    }



    return 0;
}
```

30)Write a C program to find and replace a word in a text file.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define BUFFER_SIZE 1000
```

```
/* Function declaration */
void replaceAll(char *str, const char *oldWord, const
char *newWord);



int main()
{
    /* File pointer to hold reference of input file */
    FILE * fPtr;
    FILE * fTemp;
    char path[100];

    char buffer[BUFFER_SIZE];
    char oldWord[100], newWord[100];
```

```
printf("Enter path of source file: ");
scanf("%s", path);

printf("Enter word to replace: ");
scanf("%s", oldWord);

printf("Replace '%s' with: ");
scanf("%s", newWord);



    /*  Open all required files */
fPtr  = fopen(path, "r");
fTemp = fopen("replace.tmp", "w");

    /* fopen() return NULL if unable to open file in given
mode. */
    if (fPtr == NULL || fTemp == NULL)
    {
        /* Unable to open file hence exit */
printf("\nUnable to open file.\n");
printf("Please check whether file exists and you have
read/write privilege.\n");
        exit(EXIT_SUCCESS);
    }



    /*
     * Read line from source file and write to destination
     * file after replacing given word.
     */
    while ((fgets(buffer, BUFFER_SIZE, fPtr)) != NULL)
    {
        // Replace all occurrence of word from current line
replaceAll(buffer, oldWord, newWord);
```

```
        // After replacing write it to temp file.
fputs(buffer, fTemp);
    }



    /* Close all files to release resource */
fclose(fPtr);
fclose(fTemp);



    /* Delete original source file */
    remove(path);

    /* Rename temp file as original file */
    rename("replace.tmp", path);
```

```
printf("\nSuccessfully replaced all occurrences of '%s'
with '%s'.", oldWord, newWord);


    return 0;
}




/**
 * Replace all occurrences of a given a word in string.
 */
void replaceAll(char *str, const char *oldWord, const
char *newWord)
{
    char *pos, temp[BUFFER_SIZE];
    int index = 0;
    int owlen;
```

```
owlen = strlen(oldWord);


    // Fix: If oldWord and newWord are same it goes to
infinite loop
    if (!strcmp(oldWord, newWord)) {
        return;
    }



    /*
     * Repeat till all occurrences are replaced.
     */
    while ((pos = strstr(str, oldWord)) != NULL)
    {
        // Backup current line
strcpy(temp, str);
```

```
        // Index of current found word
        index = pos - str;

        // Terminate str after word found index
        str[index] = '\0';

        // Concatenate str with new word
    strcat(str, newWord);

        // Concatenate str with remaining words after
        // oldword found index.
    strcat(str, temp + index + owlen);
      }
    }
```

31)Write a C program to replace specific line in a text file.

```
#include <stdio.h>
#include <stdlib.h>

#define BUFFER_SIZE 1000



int main()
{
    /* File pointer to hold reference of input file */
    FILE * fPtr;
    FILE * fTemp;
    char path[100];

    char buffer[BUFFER_SIZE];
    char newline[BUFFER_SIZE];
    int line, count;



printf("Enter path of source file: ");
scanf("%s", path);

printf("Enter line number to replace: ");
scanf("%d", &line);

    /* Remove extra new line character from stdin */
fflush(stdin);

printf("Replace '%d' line with: ", line);
fgets(newline, BUFFER_SIZE, stdin);



    /*  Open all required files */
```

```
fPtr  = fopen(path, "r");

fTemp = fopen("replace.tmp", "w");


    /* fopen() return NULL if unable to open file in given
mode. */

    if (fPtr == NULL || fTemp == NULL)

    {

        /* Unable to open file hence exit */

printf("\nUnable to open file.\n");

printf("Please check whether file exists and you have
read/write privilege.\n");

        exit(EXIT_SUCCESS);

    }



    /*

     * Read line from source file and write to destination
     * file after replacing given line.
     */
    count = 0;
    while ((fgets(buffer, BUFFER_SIZE, fPtr)) != NULL)
    {
        count++;

        /* If current line is line to replace */
        if (count == line)
fputs(newline, fTemp);
        else
fputs(buffer, fTemp);
    }



    /* Close all files to release resource */
fclose(fPtr);
fclose(fTemp);



    /* Delete original source file */
    remove(path);

    /* Rename temporary file as original file */
    rename("replace.tmp", path);

printf("\nSuccessfully replaced '%d' line with '%s'.", line,
newline);

    return 0;
}
```

32)Write a C program to print source code of same program.

```
#include <stdio.h>

#include <stdlib.h>



int main()

{

    FILE *fPtr;


    char ch;



    /*

     * __FILE__ is a macro that contains path of current
file.

     * Open current program in read mode.

     */

fPtr = fopen(__FILE__, "r");
```

```
    /* fopen() return NULL if unable to open file in given
mode. */

    if (fPtr == NULL)

    {

        /* Unable to open file hence exit */

printf("\nUnable to open file.\n");

printf("Please check whether file exists and you have
read privilege.\n");

        exit(EXIT_SUCCESS);

    }



    /* Read file character by character */

    while ((ch = fgetc(fPtr)) != EOF)

    {
```

```
printf("%c", ch);
    }



    /* Close files to release resources */
fclose(fPtr);

    return 0;
}
```

33)Write a C program to convert uppercase to lowercase character and vice versa in a text file.

```
#include <stdio.h>
#include <stdlib.h>

void toggleCase(FILE *fptr, const char *path);
```

```
int main()
{
    /* File pointer to hold reference of input file */
    FILE *fPtr;
    char path[100];

printf("Enter path of source file: ");
scanf("%s", path);



fPtr = fopen(path, "r");



    /* fopen() return NULL if unable to open file in given
mode. */
    if (fPtr == NULL)
    {
        /* Unable to open file hence exit */
printf("\nUnable to open file.\n");
printf("Please check whether file exists and you have
read privilege.\n");
        exit(EXIT_FAILURE);
    }



toggleCase(fPtr, path);




printf("\nSuccessfully converted characters in file from
uppercase to lowercase and vice versa.\n");


    return 0;
}
```

```
/**
 * Function to convert lowercase characters to uppercase
 * and uppercase to lowercase in a file.
 */
void toggleCase(FILE *fptr, const char *path)
{
    FILE *dest;
    char ch;

    // Temporary file to store result
dest = fopen("toggle.tmp", "w");



    // If unable to create temporary file
    if (dest == NULL)
    {
printf("Unable to toggle case.");
fclose(fptr);
        exit(EXIT_FAILURE);
    }



    /* Repeat till end of file. */
    while ( (ch = fgetc(fptr)) != EOF)
    {
        /*
         * If current character is uppercase then toggle
         * it to lowercase and vice versa.
         */
        if (isupper(ch))
ch = tolower(ch);
```

```
        else if (islower(ch))
    ch = toupper(ch);



        // Print toggled character to destination file.
    fputc(ch, dest);
      }




      /* Close all files to release resource */
    fclose(fptr);
    fclose(dest);



      /* Delete original source file */
      remove(path);
      /* Rename temporary file as original file */

      rename("toggle.tmp", path);

    }
```

34)Write a C program to find properties of a file using stat() function

```
#include <stdio.h>

#include <unistd.h>

#include <sys/stat.h>

#include <time.h>



void printFileProperties(struct stat stats);



int main()
```

```
{
    char path[100];
    struct stat stats;

printf("Enter source file path: ");
scanf("%s", path);



    // stat() returns 0 on successful operation,
    // otherwise returns -1 if unable to get file properties.
    if (stat(path, &stats) == 0)
    {
printFileProperties(stats);
    }
    else
    {
printf("Unable to get file properties.\n");
printf("Please check whether '%s' file exists.\n", path);
    }


    return 0;
}





/**
 * Function to print file properties.
 */
void printFileProperties(struct stat stats)
{
    struct tm dt;


    // File permissions
printf("\nFile access: ");
```

```
    if (stats.st_mode& R_OK)
printf("read ");
    if (stats.st_mode& W_OK)
printf("write ");
    if (stats.st_mode& X_OK)
printf("execute");

    // File size
printf("\nFile size: %d", stats.st_size);

    // Get file creation time in seconds and
    // convert seconds to date and time format
    dt = *(gmtime(&stats.st_ctime));
printf("\nCreated on: %d-%d-%d %d:%d:%d",
dt.tm_mday, dt.tm_mon, dt.tm_year + 1900,
dt.tm_hour, dt.tm_min, dt.tm_sec);
```

```
    // File modification time
    dt = *(gmtime(&stats.st_mtime));
printf("\nModified on: %d-%d-%d %d:%d:%d",
dt.tm_mday, dt.tm_mon, dt.tm_year + 1900,
dt.tm_hour, dt.tm_min, dt.tm_sec);


}
```

35)Write a C program to check if a file or directory exists.

```
#include <stdio.h>
#include <unistd.h>
#include <io.h>
#include <sys/stat.h>

int isFileExists(const char *path);
int isFileExistsAccess(const char *path);

int isFileExistsStats(const char *path);



int main()

{

    char path[100];


printf("Enter source file path: ");

scanf("%s", path);



    // Check if file exists or not

    if (isFileExistsAccess(path))

    {

printf("File exists at path '%s'\n", path);

    }
    else

    {

printf("File does not exists at path '%s'\n", path);

    }


    return 0;

}




/**

 * Function to check whether a file exists or not.

 * It returns 1 if file exists at given path otherwise

 * returns 0.

 */

int isFileExists(const char *path)

{
    // Try to open file

    FILE *fptr = fopen(path, "r");


    // If file does not exists

    if (fptr == NULL)

        return 0;


    // File exists hence close file and return true.

fclose(fptr);


    return 1;

}




/**

 * Function to check whether a file exists or not using
 * access() function. It returns 1 if file exists at
 * given path otherwise returns 0.
 */
int isFileExistsAccess(const char *path)
{
    // Check for file existence
    if (access(path, F_OK) == -1)
        return 0;

    return 1;
}




/**
 * Function to check whether a file exists or not using
 * stat() function. It returns 1 if file exists at
 * given path otherwise returns 0.
 */
int isFileExistsStats(const char *path)
{
    struct stat stats;

    stat(path, &stats);

    // Check for file existence
    if (stats.st_mode& F_OK)
        return 1;

    return 0;
}
```

36)Write a C program to rename a file using rename() function

```
#include <stdio.h>



int main()
{
    // Path to old and new files
    char oldName[100], newName[100];

    // Input old and new file name
printf("Enter old file path: ");
scanf("%s", oldName);

printf("Enter new file path: ");
scanf("%s", newName);



    // rename old file with new name
    if (rename(oldName, newName) == 0)
    {
printf("File renamed successfully.\n");
    }
    else
    {
printf("Unable to rename files. Please check files exist
and you have permissions to modify files.\n");
    }


    return 0;
}
```

37)Write a C program to list all files and sub-directories recursively.

```
#include <stdio.h>
#include <sys/types.h>
```

```
#include <dirent.h>



void listFiles(const char *path);


int main()

{

    // Directory path to list files

    char path[100];


    // Input path from user

printf("Enter path to list files: ");

scanf("%s", path);


listFiles(path);


    return 0;
```

```
}



/**
 * Lists all files and sub-directories at given path.
 */
void listFiles(const char *path)
{
    struct dirent *dp;
    DIR *dir = opendir(path);

    // Unable to open directory stream
    if (!dir)
        return;

    while ((dp = readdir(dir)) != NULL)
    {
```

```
printf("%s\n", dp->d_name);
    }

    // Close directory stream
closedir(dir);
}
```