```
//print even odd numbers in a given range
#include<stdio.h>
int Even_Odd(int);
int main()
{
int n;
printf("enter range n");
scanf("%d",&n);
Even_Odd(n);
return 0;
}

int Even_Odd(int n)
{
int even,odd;
printf("Even numbers in range are::\n");
for(even=1;even<n;even++)
{
if(even%2==0)
printf("  %d",even);
}
printf("\nOdd numbers in range are::\n");
for(odd=1;odd<n;odd++)
{
if(odd%2!=0)
printf("  %d",odd);
}
return 0;
}


//find power of a number
#include<stdio.h>
```

```
#include<math.h>
int power(int,int);
int main()
{
int num,p,res;
printf("enter number and power");
scanf("%d %d",&num,&p);
res=power(num,p);
printf("%d power %d is %d",num,p,res);
return 0;
}

int power(int num,int p)
{
int res;
res=pow(num,p);
return res;
}

//return maximum of 2 numbers
#include<stdio.h>
int max(int,int);
int main()
{
int a,b;
printf("enter 2 numbers");
scanf("%d%d",&a,&b);
max(a,b);
return 0;
}
int max(num1,num2)
{
if(num1>num2)
```

```
printf("Maximum is %d",num1);
else
printf("Maximum is %d",num2);
return 0;
}


//print all strong numbers between given interval using functions
#include<stdio.h>
int main()
{
int factorial(int);
    int fact=1,sum=0;
    int n,r;
    printf("Enter the 'n' number");
    scanf("%d",&n);
    printf("\n Strong numbers are :");
    for(int i=1;i<=n;i++)
    {
        int k=i;
        while(k!=0)
        {
            r=k%10;
            fact=factorial(r);


            k=k/10;
            sum=sum+fact;
        }
        if(sum==i){
printf("%d, ",i);

          }
```

```
        sum=0;
    }


    return 0;
}

 int factorial(int f)
    {
        int mul=1;
        for(int i=1; i<=f;i++)
        {
            mul=mul*i;
        }
        return mul;
    }


//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);
}

//Demonstrate call by value and call by reference
/*#include <stdio.h>
void swap(int , int); //prototype of the function
int main()
{
    int a = 10;
    int b = 20;
    printf("Before swapping the values in main a = %d, b = %d\
n",a,b); // printing the value of a and b in main
    swap(a,b);
    printf("After swapping values in main a = %d, b = %d\n",a,b); //
The value of actual parameters do not change by changing the
formal parameters in call by value, a = 10, b = 20
}
```

```
void swap (int a, int b)
{
    int temp;
    temp = a;
    a=b;
    b=temp;
    printf("After swapping values in function a = %d, b = %d\n",a,b); //
Formal parameters, a = 20, b = 10
}

#include <stdio.h>
void swap(int *, int *); //prototype of the function
int main()
{
    int a = 10;
    int b = 20;
    printf("Before swapping the values in main a = %d, b = %d\
n",a,b); // printing the value of a and b in main
    swap(&a,&b);
    printf("After swapping values in main a = %d, b = %d\n",a,b); //
The values of actual parameters do change in call by reference, a =
10, b = 20
}
void swap (int *a, int *b)
{
    int temp;
    temp = *a;
    *a=*b;
    *b=temp;
    printf("After swapping values in function a = %d, b = %d\n",*a,*b);
// Formal parameters, a = 20, b = 10
}
```

```c
//find power of any number using recursion
#include <stdio.h>
int power(int n1, int n2);
int main() {
    int base, a, result;
    printf("Enter base number: ");
    scanf("%d", &base);
    printf("Enter power number(positive integer): ");
    scanf("%d", &a);
    result = power(base, a);
    printf("%d^%d = %d", base, a, result);
    return 0;
}

int power(int base, int a) {
    if (a != 0)
        return (base * power(base, a - 1));
    else
        return 1;
}


//Generate fibanocii series using recursion
#include<stdio.h>
void printFibonacci(int n){
    static int n1=0,n2=1,n3;
    if(n>0){
         n3 = n1 + n2;
         n1 = n2;
         n2 = n3;
         printf("%d ",n3);
         printFibonacci(n-1);
    }
```

```
}
int main(){
    int n;
    printf("Enter the number of elements: ");
    scanf("%d",&n);
    printf("Fibonacci Series: ");
    printf("%d %d ",0,1);
    printFibonacci(n-2);//n-2 because 2 numbers are already printed
  return 0;
 }

//find product of two numbers using recursion
#include <stdio.h>
#include <stdlib.h>

int product(int,int);   //function prototype / declaration
int main()
{
    int num1,num2,result;  //variable declarataion
    printf("Enter two number to find their product\n");
    scanf("%d %d",&num1,&num2);  //numbers receive from the user
    result=product(num1,num2);//assign the output to variable result
    //function call
    printf("PRoduct of %d and %d is %d\n",num1,num2,result);

    return 0;
}

int product(int a, int b)  //function definition
{
    if(a<b)
    {
        return product(b,a);
```

```
    }
else if(b!=0){
            return (a+product(a,b-1));

    }
    else{
        return 0;
    }
}

//find sum of digits of a number. number must be passed to function
using pointers
#include<stdio.h>
int sum_of_digits(int*);
int main()
{
int n;
printf("Enter a number:");
scanf("%d",&n);
sum_of_digits(&n);
return 0;
}

int sum_of_digits(int *p)
{
int m,sum=0;
while(*p>0)
{
m=*p%10;
sum=sum+m;
*p=*p/10;
}
printf("Sum is=%d",sum);
```

```
return 0;
}

//GCD of two numbers using recursion
#include <stdio.h>
int hcf(int n1, int n2);
int main() {
    int n1, n2;
    printf("Enter two positive integers: ");
    scanf("%d %d", &n1, &n2);
    printf("G.C.D of %d and %d is %d.", n1, n2, hcf(n1, n2));
    return 0;
}

int hcf(int n1, int n2) {
    if (n2 != 0)
        return hcf(n2, n1 % n2);
    else
        return n1;
}

//lcm of two numbers using recursion
#include <stdio.h>

int lcm(int, int);

int main()
{
    int a, b, result;
    int prime[100];

    printf("Enter two numbers: ");
    scanf("%d%d", &a, &b);
```

```
    result = lcm(a, b);
    printf("The LCM of %d and %d is %d\n", a, b, result);
    return 0;
}

int lcm(int a, int b)
{
    static int common = 1;

    if (common % a == 0 && common % b == 0)
    {
return common;
    }
    common++;
    lcm(a, b);
    return common;
}
```