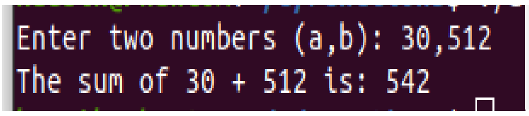
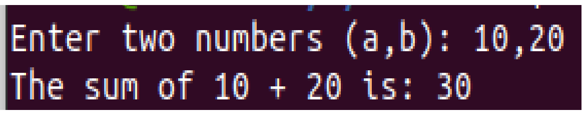
1. Create a function named “addNumbers” that accepts two numbers and finds their sum

```
#include<stdio.h>

// user defined function to add two numbers
int addNumbers(int n1, int n2)
{
	int sum;
	sum = n1 + n2;
	return sum;
}
int main()
{
	int a, b, add;   //variable declaration
	printf("Enter two numbers (a,b): "); //Accepts two numbers
	scanf("%d,%d", &a, &b);

//Assigns the value of function to variable add
	add = addNumbers(a,b);
	printf("The sum of %d + %d is: %d\n", a, b, add); //prints result
	return 0;
}
```



```
Enter two numbers (a,b): 30,512
The sum of 30 + 512 is: 542
```



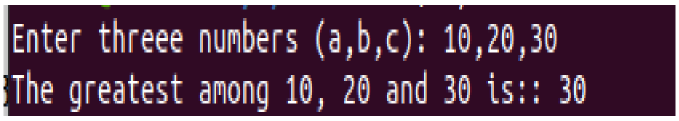
```
Enter two numbers (a,b): 10,20
The sum of 10 + 20 is: 30
```

This is a program that returns the sum of two integer numbers with the use of user defined function addNumbes(). This function accepts two user input values and returns integer as the sum of two input numbers.

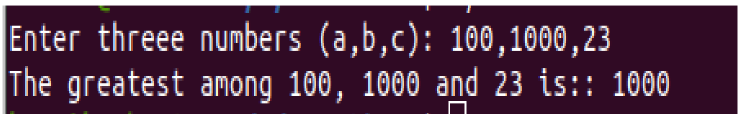
2. Using the concept of function, write a program to find the greatest number among 3 numbers.

```
#include<stdio.h>

//function declaration with three arguments (n1, n2, n3)
int greatest(int n1, int n2, int n3)
{
	if (n1 > n2 && n1 > n3)
		return n1;
	if (n2 > n1 && n2 > n3)
		return n2;
	if (n3 > n1 && n3 > n2)
		return n3;
}
int main()
{
	int a, b, c, great; //variable declaration
	printf("Enter threee numbers (a,b,c): ");  // Accepts input
	scanf("%d %d %d", &a, &b, &c);
	printf(“Your input: %d, %d, %d\n”, a, b, c);

//Assigns the value of function to variable great
	great = greatest(a,b,c);
	printf("The greatest among is:: %d\n", great);
```

```
          return 0;
     }
```



```
Enter threee numbers (a,b,c): 10,20,30
The greatest among 10, 20 and 30 is:: 30
```

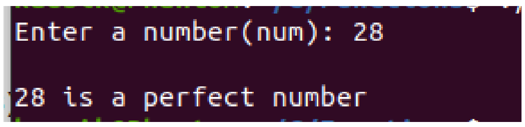


```
Enter threee numbers (a,b,c): 100,1000,23
The greatest among 100, 1000 and 23 is:: 1000
```

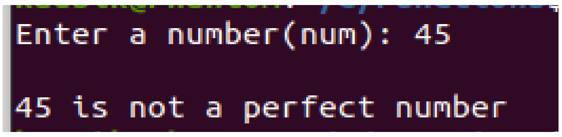
This is a program that returns the greatest among three integer numbers with the use of function greatest(). This program accepts three user input values and returns the greatest integer among three input numbers.

3. Using functions, check whether the number is a perfect number or not. The Perfect number will be equal to the sum of all its positive divisors below of the number.

```
#include<stdio.h>

//function declaration with one integer arguement
int perfectNum(int n1){
     int i, sum = 0 ;            //variable declaration
     for(i = 1; i < n1; i++){
          if (n1 % i == 0){
               sum+=i;
          }
     }
     return sum;                 //returns the value of sum
}
int main(){
     int num, a;                 //variable declaration
     printf("Enter a number(num): "); //accepts input
     scanf("%d", &num);

//Assigns value of function perfectNum() to variable a
     a = perfectNum(num);
     if(a == num){          //checks if value of a equals num
          printf("\n%d is a perfect number\n", num);
     }
     else{
          printf("\n%d is not a perfect number\n", num);
     }
     return 0;
}
```



```
Enter a number(num): 28

28 is a perfect number
```

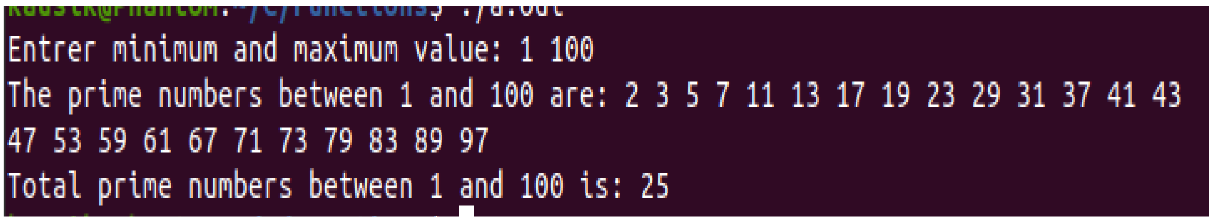


```
Enter a number(num): 45

45 is not a perfect number
```

This is a program that accepts a input from user and checks whether or not it is perfect number. The function perfectNum() returns the sum of factors of input number excluding the input number. The return value of function is stored in variable ‘a’ which is compared with original number and the output is displayed accordingly

4. Write a program to find all the prime numbers between 1 to 100.

```c
#include<stdio.h>

//function declaration with one integer argument
int isPrime(int num)
{
     int p;                     //variable declaration
     for (p = 2 ; p < num ; p++){
                if (num % p == 0) //p is not prime
                     return 0;
     }
//p is prime, so it is returned
     return 1;
}

int main(){
     int min, max, count , i = 0;    //variable declaration

//Accepting minimum and maximum values
     printf("Entrer minimum and maximum value: ");
     scanf("%d%d", &min, &max);
     printf("The prime numbers between %d and %d are: ", min ,max);
     for(count = min+1; count < max; count++){

//if returned value is true (i.e. 1), then it is printed.
           if (isPrime(count)){
                printf("%d ", count);
                i++;            //count of prime is increased by 1.
           }
     }
     printf("\n");

//displays the total number of prime between two input values
     printf("Total prime numbers between %d and %d is: %d\n", min, max, i);
     return 0;
}
```

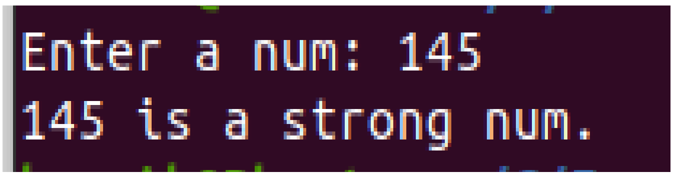


```
Entrer minimum and maximum value: 1 100
The prime numbers between 1 and 100 are: 2 3 5 7 11 13 17 19 23 29 31 37 41 43
47 53 59 61 67 71 73 79 83 89 97
Total prime numbers between 1 and 100 is: 25
```

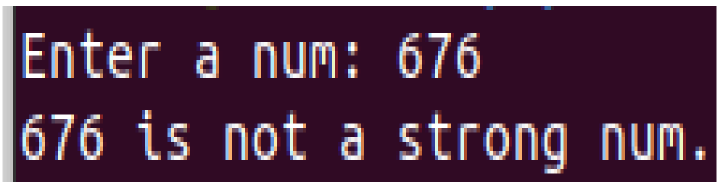
This is a program that asks user to input a minimum and a maximum value and displays the prime numbers between them. Finally, the total count of prime numbers between the range is displayed. The function isPrime() takes in argument ‘num’ which iterates from (min + 1) to (max – 1). ‘p’ is the variable to be checked for prime. If it satisfies the condition of prime number, its value is returned by the function and is displayed as output.

5. Write a program to check whether a number is strong or not.

```
#include<stdio.h>
//factorial of num
int factorial(int num){
     if (num == 1 || num == 0)
          return 1;
     if (num > 1)
          return num*factorial(num-1);

}
//function to check if the number is strong or not
int strongNum(int num){
     int store, rem, value = 0;      //variable declaration

//store stores the value of num so that it doesn't change after looping
     store = num;
     while(num > 0 && store != 0)
     {
          rem = store % 10;
          value += (factorial(rem));
          store /= 10;

     }
//checking if value is equal to num
     if (value == num)
//if condition is true return the value of 'true = 1' from this function
          return 1;
     else
          return 0;
}
int main(){

     int n, a;                    //variable declaration
     printf("Enter a num: ");     //asking user input
     scanf("%d", &n);

//a has the value of function strongNum
     a = strongNum(n);

//if returned value is true (i.e. 1) then number is strong
     if (a)
          printf("%d is a strong num.\n", n);
     else
          printf("%d is not a strong num.\n", n);

     return 0;
}
```



```
Enter a num: 145
145 is a strong num.
```



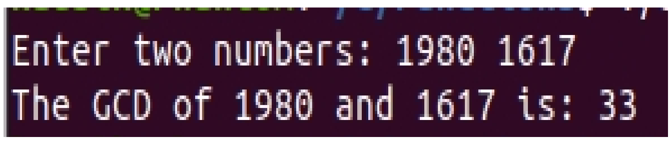
```
Enter a num: 676
676 is not a strong num.
```

This is a program that asks a input number from user and checks whether it is a strong number or not using function and displays the output. A **strong number** is such a number whose sum of factorials of individual digit is equal to the number itself.
For eg: 145 is a strong number because 1! + 4! + 5! = 145
In this program, a separate factorial(int num) function is created to compute the factorial of individual digits of input number recursively.
As the factorial of individual digit is required, it is a lengthy process to perform a repititive task. Thus a function is created and is called when needed in the program.

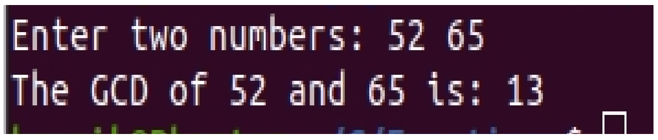
In the function strongNum(), the input number is broken down into individual digit and sum of factorial of individual digit is performed. If the input number and the value obtained after performing the operation is equal then the value(**=true i.e. 1)** is returned from the function and output is displayed accordingly.

6. Write a program to evaluate the GCD of two numbers using functions or recursively find the GCD of two input numbers.

```
#include<stdio.h>

int gcd(int n1, int n2){

      if (n1 == 0)
            return n2;
      if (n2 == 0)
            return n1;
      if (n1 > n2)
            return gcd(n1 % n2, n2);
      if (n2 > n1)
            return gcd(n1, n2 % n1);



}

int main(){
      int a, b, g;                          //variable declaration
      printf("Enter two numbers: ");   //accepts user input
      scanf("%d %d", &a, &b);

//Assigns the value of function to variable g
      g = gcd(a,b);
      printf("The GCD of %d and %d is: %d\n", a, b, g); //displays result
      return 0;
}
```



```
Enter two numbers: 1980 1617
The GCD of 1980 and 1617 is: 33
```



```
Enter two numbers: 52 65
The GCD of 52 and 65 is: 13
```

Recursion is a process in which a function calls itself within the same function until a specific condition is satisfied. In recursion, all the partial solutions are combined to give a final solution. This program accepts two input numbers ‘a’ and ‘b’ and displays the GCD(Greatest Common Divisor) of the two recursively using the function gcd(). The base condition for exiting of function is either n1 or n2 is euqal to zero.

Let’s breakdown the function gcd(int n1. int n2) to see how it works:

Consider n1 = 1980 and n2 = 1617 as two arguments passed into function.
Both n1 and n2 are not equal to zero and n1 > n2 so the function returns
gcd (n1 % n2, n2)
i.e. gcd (1980 % 1617, 1617) = gcd (363, 1617)
thus new value of n1 = 363, n2 = 1617

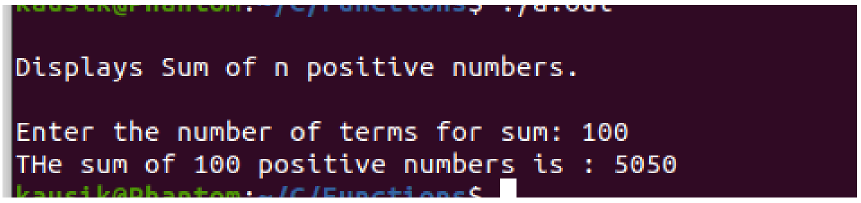
here n2 > n1
so gcd(n2 % n1, n1)
i.e. gcd (363, 1617 % 363) = gcd (363, 165) is returned.

Again new n1 = 363 and n2 = 165
the process is repeated until n1 = 33 and n2 = 0

Thus when n2 = 0, the value of n1 = 33 is returned, which is the GCD of two input numbers 1980 and 1617.
This method of finding the GCD of two numbers is called Eucledian’s Algorithm.

7. Using the concept of recursion, find the sum of n positive numbers. *

```
#include<stdio.h>

//function prototyping
     int sumNumbers(int n);

     int main(){
          int n, sum;                //variable declaration
          printf("\nDisplays Sum of n positive numbers.\n\n");

          printf("Enter the number of terms for sum: "); //Accepts input
          scanf("%d", &n);

//Assigns the value of function to variable sum
          sum = sumNumbers(n);
          printf("THe sum of %d positive numbers is : %d\n", n, sum);
          return 0;
     }

     int sumNumbers(int n){
     if (n >= 0)
          return n+sumNumbers(n-1);
     if (n < 0)
          return 0;
     }
```



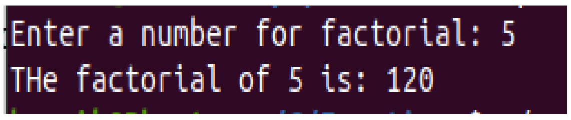
```
Displays Sum of n positive numbers.

Enter the number of terms for sum: 100
THe sum of 100 positive numbers is : 5050
kausik@Phantom:~/C/Functions$
```

This is a program that accepts a user input positive integer and displays the continuous sum of numbers from 1 to itself recursively using function sumNumbers(). The function sumNumbers() accepts user input ‘n’ and computes sum until n equals to 0. If n equals 0, then the function returns the value of sum of numbers upto ‘n’ and displays the output.

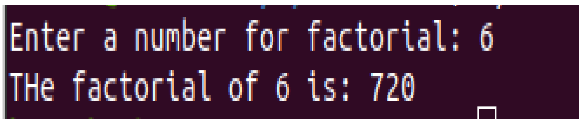
The function sumNumbers(int n) is defined before main() function so that the compiler is made aware about the use of function later in the program. This technique of declaration of function before the main() function is called function prototyping.

8. Recursively, find the factorial of an input number n.

```
#include<stdio.h>

//function prototyping
int factorial(int n);

int main(){
     int n, fact;          //variable declaration
     printf("Enter a number for factorial: ");  //Accepts input
     scanf("%d", &n);

//assigns value of function to variable fact
     fact = factorial(n);
     printf("The factorial of %d is: %d\n", n, fact); //Displays result
     return 0;

}

int factorial(int n)
{
     if (n == 0 || n == 1)
          return 1;
     if (n > 1)
          return n*factorial(n-1);
     if (n < 0)
          return 0;

}
```



```
Enter a number for factorial: 5
THe factorial of 5 is: 120
```



```
Enter a number for factorial: 6
THe factorial of 6 is: 720
```

This is a program that displays the factorial of a given number recursively. It accepts a integer input from user and computes factorial using the function factorial(). The function factorial() takes in the user input (n) and computes factorial until ‘n’ equals to zero. If n equals 0 then the program exits by returning the factorial of the input number. If the input is 0 or 1, the return value is 1(i.e. true). If the input is less than 0, the program exits with return value 0 (i.e. false).

The function factorial(int n) is defined before main() function so that the compiler is made aware about the use of function later in the program. This technique of declaration of function before the main() function is called function prototyping.