1. **Write a C program to check whether a number is palindrome or not**

An integer is a **palindrome** when it reads the same backward as forward. For example, `121` is a palindrome while `123` is not.

Step1: Read the number n and copy the value in another variable
i.e. number=n
Step2: reverse the number n//if n=235 so reverse will be 532

```
number = n;
reverse=0;
while( n!=0 )
begin
        remainder = n%10;
        reverse = reverse*10 + remainder;
        n /= 10;
end
```

Step 3:if (number == reverse) then n is palindrome else it is not palindrome.

2. **Write a C program to find all factors of a number.**

A number b is factor of a number a if a is divisible by b,i.e. if(a%b==0) then b is the factor of a.

Step1: Read the number n
Step2: For i in 1 to n
Begin
If a %i equal to zero then print i is factor of n
End.

3. **Write a C program to calculate factorial of a number**

Product of all consecutive Integer numbers up to n is called Factorial of a Number and is denoted by n! For Example, the value of 5! is 120.

Mathematically it is written as,
n! = 1 * 2 * 3 * 4 * ... * (n-1) * n
For example, the factorial of 5 is,
5! = 1 * 2 * 3 * 4 * 5 = 120

Step 1: Start
Step 2: Declare Variable n, fact, i
Step 3: Read number from User
Step 4: Initialize Variable fact=1 and i=1
Step 5: while i<=number
5.1 fact=fact*i
5.2 i=i+1
Step 6: Print fact
Step 7: Stop

4. **Write a C program to find HCF (GCD) of two numbers.**

GCD stands for Greatest Common Divisor. So GCD of 2 numbers is nothing but the largest number that divides both of them.
Example: Lets say 2 numbers are 36 and 60. Then
36 = 2*2*3*3
60 = 2*2*3*5
GCD=2*2*3
i.e GCD=12

GCD is also known as HCF (Highest Common Factor)

Algorithm for Finding GCD of 2 numbers:
Step 1: Start
Step 2: Declare variable n1, n2, gcd=1, i=1
Step 3: Input n1 and n2
Step 4: Repeat until i<=n1 and i<=n2
Step 4.1: If n1%i==0 && n2%i==0:
Step 4.2: gcd = i
Step 5: Print gcd
Step 6: Stop

5. **Write a C program to check whether a number is Prime number or not.**

A number that's only divisible by 1 and itself is named a Prime Number. For Example, 3, 5, 7, 11 are Prime Numbers.
Note: 2 is the only even prime number.
Step 1: Start
Step 2: Initialize variables num,flag=1, j=2
Step 3: Read num from user
Step 4: If num<=1 // Any number less than 1 is not a prime number
Display "num is not a prime number"
Goto step 7
Step 5: Repeat the steps until j<[(n/2)+1]
5.1 If remainder of number divide j equals to 0,
Set flag=0
Goto step 6
5.2 j=j+1
Step 6: If flag==0,
Display num+" is not prime number"
Else
Display num+" n is prime number"
Step 7: Stop

6. **Write a C program to check whether a number is Armstrong number or not**.

What is an Armstrong number?

An Integer number in which the sum of the power of number of digits is same as the number is called as Armstrong Number
for a three digit number the sum of the cubes of its digits is equal to the number itself For example, 153 is an Armstrong number since 1**3 + 5**3 + 3**3 = 153.
Algorithm to check whether a 3-digit number is Armstrong or not
Step 1: Start
Step 2: Declare Variable sum, temp, num
Step 3: Read num from User
Step 4: Initialize Variable sum=0 and temp=num
Step 5: Repeat Until num>=0
    5.1 sum=sum + cube of last digit i.e [(num%10)*(num%10)*(num%10)]
    5.2 num=num/10
Step 6: IF sum==temp
    Print "Armstrong Number"
  ELSE
    Print "Not Armstrong Number"
Step 7: Stop

7. **Write a C program to check whether a number is Perfect number or not**.

What is Perfect Number?
Perfect number is a positive integer equal to the sum of its proper divisors. Sum of its proper divisor excludes the Number itself. Ex. For number 6, the divisors are 1, 2, 3 and 6. Now if we take sum of 1, 2, 3 and exclude the number itself (i.e. 6), the sum is 6. Hence, 6 is a perfect number. 6 is the smallest Perfect Number.
1. Start
2. Read n
3. Initialize s=0
4. for i=1 to n do
    a. if(n%i)==0, then
    b. s=s+i
5. if s==n
    then Print "Given Number is Perfect Number". Goto Step 7
6. Print "Given Number is Not a Perfect Number"
7. Stop

Strong number is a number whose sum of all digits' factorial is equal to the number 'n'. Factorial implies when we find the product of all the numbers below that number including that number and is denoted by ! (Exclamation sign), For example: 4! = 4x3x2x1 = 24.

So, to find a number whether its strong number, we have to pick every digit of the number like the number is 145 then we have to pick 1, 4 and 5 now we will find factorial of each number i.e, 1! = 1, 4! = 24, 5! = 120.

Now we will sum up 1 + 24 + 120 so we get 145, that is exactly same as the input given, So we can say that the number is strong number.

```
START
In Function int factorial(int r)
  Step1 -> Initialize int fact and set as 1
  Step2-> Loop while r>1
    Set fact as fact * r
    Decremnet r by 1
  End Loop
  Step 3-> Return fact
  End Function factorial
In Function int check(int n)
  Step 1-> Initialize int temp, rem and result, set result as 0
  Step 2-> Set temp as n
  Step 3-> Loop while temp
    Set rem as temp % 10
    Set result as result + factorial(rem)
    Set temp as temp/10
  End loop
  Step 4-> If result == n then,
    Return 1
  Step 5-> Else
  Return 0
  End function check
In main(int argc, char const *argv[])
  Step 1-> Initialise and set n as 145
  Step 2->If check(n) is valid then,
    Print "Yes it is a strong number”
  Step 3-> Else
    Print "no it is not a strong number”
STOP
```

8. **Write a C program to print Fibonacci series up to n terms.**

A series of numbers in which each number is the sum of the two preceding or previous numbers is called Fibonacci Series.

For example, Fibonacci series upto 7 numbers is 1, 1, 2, 3, 5, 8, 13.
In above example, first 2 numbers (1, 1) are printed directly as there are no preceding numbers. But after that, i.e the 3rd number (2) is the sum of 1st and 2nd number (1+1=2). Similarly to get 4th number, we add 2nd and 3rd number. (i.e., 1+2=3). You can use this pattern to find fibonacci series upto any number.

Mathematical expression to find Fibonacci number is :

Fn=Fn-1+Fn-2

i.e. To get nth position number, you should add (n-2) and (n-1) position number.

Step 1: Start
Step 2: Declare variable a, b, c, n, i
Step 3: Initialize variable a=0, b=1 and i=2
Step 4: Read n from user
Step 5: Print a and b
Step 6: Repeat until i<=n :
    Step 6.1: c=a+b
    Step 6.2: print c
    Step 6.3: a=b, b=c
    Step 6.4: i=i+1
Step 7: Stop