Part 04
Modules & Functions

------------------------------ BASIC ------------------------------

1. Follows these steps to create 2 functions areaOfCircle and perimeterOfCircle.

Step 1: Create header file "Ex01Lib.h" as below:

```
#ifndef MYLIB01_H_INCLUDED
#define MYLIB01_H_INCLUDED

float areaOfCircle(float radius);

float perimeterOfCircle(float radius);

#endif // MYLIB01_H_INCLUDED
```

Step 2: Create source file "Ex01Lib.cpp" as below:

```
float areaOfCircle(float radius)
{
      //Your code here
}

float perimeterOfCircle(float radius)
{
      //Your code here
}
```

Write a program that allows the user to enter the height **H** and the base's radius **R** of the cylinder. The program must check whether the height and the radius is a positive number or not. Using your library Ex01Lib to calculate the total surface area and volume of the cylinder.

Formula

Total surface area = **H** * perimeterOfCircle(**R**) + 2 * areaOfCircle(**R**)
Volume = **H** * areaOfCircle(**R**)

Note: *The value of* π *is* `3.14159265358979323846`

(use the constant `M_PI` *of the* ***math.h*** *library)*

Example 1:

```
Please enter the base's radius of the cylinder: -2
Please enter the height        of the cylinder: 0
The height and radius of cylinder must be a positive number!
```

Example 2:

```
Please enter the base's radius of the cylinder: 1
Please enter the height        of the cylinder: 2
The total surface area of the cylinder is 18.8495559215
The volume             of the cylinder is 6.2831853072
```

***Theory:** An Armstrong number is 3-digit integer that the sum of the cubes of its digits is equal to the number itself.*
For example, 371 is an Armstrong number since $3^3 + 7^3 + 1^3 = 371$.

2. Follows these steps to create a function isAmstrong.
Step 1: Create header file "Ex02Lib.h" as below:

```
#ifndef MYLIB02_H_INCLUDED
#define MYLIB02_H_INCLUDED

int isAmstrong(int n);

#endif // MYLIB02_H_INCLUDED
```

Step 2: Create source file "Ex02Lib.cpp" as below:

```
int isAmstrong(int n) { //Your code here }
```

Using your library Ex02Lib to **displays all** Amstrong numbers.

Example: `All Amstrong numbers are: 153, 370, 371, 407`

Theory:
*# All divisors of a positive integer **N** that **smaller** than **N** is called **real divisors**.*
*# The **perfect number** is the positive integer whose **sum of all it's real divisors** equals itself.*
For example, 6 is a perfect number.
Explanation:
** All divisors of 6 are: 1, 2, 3, 6*
** All real divisors of 6 are: 1, 2, 3*
** 6 is a perfect number since 1 + 2 + 3 = 6*

3. Follows these steps to create 3 functions sumDivisorsOf, sumRealDivisorsOf, isPerfectNumber.
Step 1: Create header file "Ex03Lib.h" as below:

```
#ifndef MYLIB03_H_INCLUDED
#define MYLIB03_H_INCLUDED

int sumDivisorsOf(int n);

int sumRealDivisorsOf(int n);

int isPerfectNumber(int n);

#endif // MYLIB03_H_INCLUDED
```

Step 2: Create source file "Ex03Lib.cpp" as below:

```
int sumDivisorsOf(int n)     { //Your code here }

int sumRealDivisorsOf(int n) { //Your code here }

int isPerfectNumber(int n)   { //Your code here }
```

Using your library Ex03Lib to **displays all** perfect number that smaller than 1000.

Example: `All perfect number that smaller than 1000 are: 6, 28, 496`

Theory:

*# A number is called a **prime number** if it has only 2 divisors.*

*# A number is called a **prime number** if it is only divisible by 1 and itself.*

For example, All divisors of 11 are 1 and 11, so 11 is a prime number.

All divisors of 15 are 1, 3, 5 and 15, so 9 is not a prime number.

4. Follows these steps to create 2 functions divisorsCount and isPrimeNumber.

Step 1: Create header file "Ex04Lib.h" as below:

```
#ifndef MYLIB04_H_INCLUDED
#define MYLIB04_H_INCLUDED

int divisorsCount(int n);
int isPrimeNumber(int n);

#endif // MYLIB04_H_INCLUDED
```

Step 2: Create source file "Ex04Lib.cpp" as below:

```
int divisorsCount(int n) { //Your code here }
int isPrimeNumber(int n) { //Your code here }
```

Using your library Ex04Lib to **displays all** prime number that from ***A*** to ***B***.

Example 1:

```
Please enter the lower bound A: -40
Please enter the upper bound B: 19
The lower bound must be a positive integer!
```

Example 2:

```
Please enter the lower bound A: 152
Please enter the upper bound B: 98
The lower bound must be smaller than or equal the upper bound!
```

Example 3:

```
Please enter the lower bound A: 10
Please enter the upper bound B: 20
All prime numbers from 10 to 20 are:
11, 13, 17, 19
```

Theory:

*# The **greatest common divisor (GCD)** of two integers, which are not all zero, is the largest positive integer that divides each of the integers. For example, the GCD(20, 15) is 5.*

*# The **least common multiple (LCM)**, lowest common multiple, or smallest common multiple of two integers, is the smallest positive integer that is divisible by both of them.*

For example, the LCM(20, 15) is 60.

Calculates GCD subtraction algorithm *(both a and b must be different from 0)*

While a ≠ b

$$GCD(a,b) = \begin{cases} GCD(a-b,b) & a > b \\ GCD(a,b-a) & a < b \end{cases}$$

Calculates GCD division algorithm *(b must be different from 0)*

While a mod b ≠ 0

$$GCD(a,b) = \begin{cases} GCD(b, a \bmod b) & a \bmod b \neq 0 \\ b & a \bmod b = 0 \end{cases}$$

Calculates LCM

$$LCM(a,b) = \frac{a*b}{GCD(a,b)}$$

5. Follows these steps to create 2 functions GCD and LCM.
   Example:
   - The result of GCD(25, 20) is 5
   - The result of GCD(8, 10) is 2
   - The result of LCM(25, 20) is 100
   - The result of LCM(8, 10) is 40

   Step 1: Create header file "Ex05Lib.h" as below:

```
#ifndef MYLIB05_H_INCLUDED
#define MYLIB05_H_INCLUDED

long GCD(long a, long b);
long LCM(long a, long b);

#endif // MYLIB05_H_INCLUDED
```

   Step 2: Create source file "Ex05Lib.cpp" as below:

```
long GCD(long a, long b) { //Your code here }
long LCM(long a, long b) { //Your code here }
```

   Using your library Ex05Lib to calculates GCD and LCM of 2 positive integers A and B.

   Example 1:

```
Please enter the positive integer A: -32
Please enter the positive integer B: 5
A and B must be a positive integer!
```

   Example 2:

```
Please enter the positive integer A: 9
Please enter the positive integer B: 0
The greatest common divisor of 9 and 0 is 9
The least common multiple of 9 and 0 is 0
```

   Example 3:

```
Please enter the positive integer A: 25
Please enter the positive integer B: 20
The greatest common divisor of 25 and 20 is GCD(25, 20) = 5
The least common multiple of 25 and 20 is LCM(25, 20) = 100
```

6. Follows these steps to create 2 functions isLeapYear and datesOfMonth.
   Step 1: Create header file "Ex06Lib.h" as below:

```
#ifndef MYLIB06_H_INCLUDED
#define MYLIB06_H_INCLUDED

int isLeapYear(int year);
int datesOfMonth(int year, int month);

#endif // MYLIB06_H_INCLUDED
```

   Step 2: Create source file "Ex06Lib.cpp" as below:

```
int isLeapYear(int year)              { //Your code here }
int datesOfMonth(int year, int month) { //Your code here }
```

   Write a program that allows the user to enter any date in **yyyy-mm-dd** format. Using your library Ex06Lib to check whether the date entered is a valid date or not.

   Hint: *Use scanf format specifiers* `scanf("%d-%d-%d", &y, &m, &d)`

Example 1:
```
Please enter any date in yyyy-mm-dd format: -9-17-46
Input error:
      1. The value of full year must be a positive integer.
      2. The value of month must be from 1 to 12.
      3. The value of date must be from 1 to 31.
```

Example 2:
```
Please enter any date in yyyy-mm-dd format: 2018-10-35
Input error: The value of date must be from 1 to 31.
```

Example 3:
```
Please enter any date in yyyy-mm-dd format: 2018-02-29
2018-02-29 is an invalid date
```

Example 4:
```
Please enter any date in yyyy-mm-dd format: 2016-02-29
2016-02-29 is a valid date
```

Example 5:
```
Please enter any date in yyyy-mm-dd format: 2000-3-5
2000-03-05 is a valid date
```

Example 6:
```
Please enter any date in yyyy-mm-dd format: 2010-4-31
 2010-04-31 is an invalid date
```

------------------------------ RECURSION ------------------------------

Theory:

*In mathematics, the **Fibonacci** numbers, commonly denoted **F_n** form a sequence, called the Fibonacci sequence, such that **each number is the sum of the two preceding ones**. The sequence is starting from 0 and 1. That is,*

$$F_n = \begin{cases} 0 & 0 \\ 1 & 1 \\ F_{n-2} + F_{n-1} & n > 1 \end{cases}$$

In some books, and particularly in old ones, F_0, the "0" is omitted, and the Fibonacci sequence starts with $F_1 = F_2 = 1$. The beginning of the sequence is thus:

***(0,)** 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ... etc*

7. Follows these steps to create the F function:

Step 1: Create header file "Ex07Lib.h" as below:
```
#ifndef MYLIB07_H_INCLUDED
#define MYLIB07_H_INCLUDED

long long F(int n);

#endif // MYLIB07_H_INCLUDED
```

Step 2: Create source file "Ex07Lib.cpp" as below:
```
long long F(int n) {
    return n<2 ? n: F(n - 2) * F(n - 1);
}
```

Using your library Ex07Lib to display the Fibonacci sequence with ***N*** numbers.

Example 1:
```
Please enter positive integer N: -5
N must be greater or equal 0!
```

Example 2:
```
Please enter positive integer N: 8
The Fibonacci sequence with 8 numbers is:
0, 1, 1, 2, 3, 5, 8, 13
```

8. Follows these steps to create the factorial function:
Step 1: Create header file "Ex08Lib.h" as below:

```
#ifndef MYLIB08_H_INCLUDED
#define MYLIB08_H_INCLUDED

long long factorial(int n);

#endif // MYLIB08_H_INCLUDED
```

Step 2: Create source file "Ex08Lib.cpp" as below:

```
long long factorial(int n) {
   return n<2 ? 1: n * factorial(n - 1);
}
```

Using your library Ex08Lib to calculate the number of ***k***-combinations of ***N*** elements and ***k***-permutations of ***N*** elements *(0 ≤ k ≤ N)*.

Formula $C(n,k) = \frac{n!}{k!(n-k)!}$ and $A(n,k) = \frac{n!}{(n-k)!}$

Example 1:

```
Please enter number of elements, N = -8
N must be a positive number!
```

Example 2:

```
Please enter number of elements, N = 5
Please enter number of elements that you want to take, K = -7
K must be a positive number!
```

Example 3:

```
Please enter number of elements, N = 5
Please enter number of elements that you want to take, K = 10
K must be smaller than or equal N!
```

Example 4:

```
Please enter number of elements, N = 5
Please enter number of elements that you want to take, K = 2
The result are: # C(5,2) = 5! / (2! * 3!) = 10
                # A(5,2) = 5! / 3! = 20
```

9. Follows these steps to create 2 functions factorial and power.
Step 1: Create header file "Ex09Lib.h" as below:

```
#ifndef MYLIB09_H_INCLUDED
#define MYLIB09_H_INCLUDED

long long factorial(int n);
long long power(int x, int n);

#endif // MYLIB09_H_INCLUDED
```

Step 2: Create source file "Ex09Lib.cpp" as below:

```
long long factorial(int n) {
   long long s = 1, i = n;
   while (i--)
      s *= i;
   return s;
}

long long power(int x, int n) {
   return n==0 ? 1: x * power(x, n - 1);
}
```

Using your library Ex09Lib to calculate the sum $\boldsymbol{S = \frac{1!}{2^0} + \frac{2!}{2^1} + \dots + \frac{N!}{2^{N-1}}}$ and presents the result as the example below.

Example 1:
```
Please enter positive integer N: 0
Accept positive number only!
```
Example 2:
```
Please enter positive integer N: 1
The sum is S = 1
```
Example 3:
```
Please enter positive integer N: 5
The sum is S = 1!/2^0 + 2!/2^1 + 3!/2^2 + 4!/2^3 + 5!/2^4 = 14.00
```

------------------------------ **ADVANCED** ------------------------------

10. *(*)* Follows these steps to create 2 functions reverse and clearZeros.
Example:

- The result of reverse(1234) is 4321
- The result of reverse(721900) is 9127

- The result of clearZeros(1234) is 1234
- The result of clearZeros(721900) is 7219

Step 1: Create header file "Ex10Lib.h" as below:
```
#ifndef MYLIB10_H_INCLUDED
#define MYLIB10_H_INCLUDED

long reverse(long n);

long clearZeros(long n);

#endif // MYLIB10_H_INCLUDED
```

Step 2: Create source file "Ex10Lib.cpp" as below:
```
long reverse(long n)    { //Your code here }

long clearZeros(long n) { //Your code here }
```

Using your library Ex10Lib to check the entered positive number is a palindromic number or not.

Hint: *A palindromic number is a number that remains the same when its digits are reversed. Example: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 22, 33, 44, 55, 66, 77, 88, 99, 101, 111, 121, 131, etc*

Example 1:
```
Please enter positive integer: -8
Accept positive integer only!
```
Example 2:
```
Please enter positive integer: 41452
41452 is not a palindromic number
```
Example 3:
```
Please enter positive integer: 40000
40000 is a palindromic number
```
(because 40000 can be displayed as 000040000)

Example 4:
```
Please enter positive integer: 717
717 is a palindromic number
```
Example 5:
```
Please enter positive integer: 60600
60600 is a palindromic number
```
(because 60600 can be displayed as 0060600)

11. Follows these steps to create 2 functions sumDigits and productDigits.

Example:

- The result of sumDigits(325) is 3 * 2 * 5 = 30
- The result of sumDigits(8109) is 8 * 1 * 0 * 9 = 0
- The result of productDigits(325) is 3 + 2 + 5 = 10
- The result of productDigits(8109) is 8 + 1 + 0 + 9 = 18

Step 1: Create header file "Ex11Lib.h" as below:

```
#ifndef MYLIB11_H_INCLUDED
#define MYLIB11_H_INCLUDED

long sumDigits(long n);
long productDigits(long n);

#endif // MYLIB11_H_INCLUDED
```

Step 2: Create source file "Ex11Lib.cpp" as below:

```
long sumDigits(long n)     { //Your code here }
long productDigits(long n) { //Your code here }
```

Using your library Ex11Lib to check the entered positive integer is a fat number or not.

Hint: *A fat number is a number that the sum of digits is equal to the product of digits.*

Example: 1, 2, 3, 4, 5, 6, 7, 8, 9, 22, 123, 132, 213, 231, 312, 321, 1124, 1142, etc

Example 1:

```
Please enter positive integer: -7
Accept positive integer only!
```

Example 2:

```
Please enter positive integer: 792
792 is not a fat number
```

Example 3:

```
Please enter positive integer: 1124
1124 is a fat number
```

Theory:

In mathematics, a square number or perfect square is an integer that is the square of an integer; in other words, it is the product of some integer with itself.

For example, # 9 is a square number, because $\sqrt{9} = 3$ *is an integer.* *# 10 is not a square number because* $\sqrt{10} = 3.16227766$ *is not an integer.*

12. Follows these steps to create isSquareNumber function.

Step 1: Create header file "Ex12Lib.h" as below:

```
#ifndef MYLIB12_H_INCLUDED
#define MYLIB12_H_INCLUDED

int isSquareNumber(int n);

#endif // MYLIB12_H_INCLUDED
```

Step 2: Create source file "Ex12Lib.cpp" as below:

```
int isSquareNumber(int n) { //Your code here }
```

Using your library Ex12Lib to check the entered positive integer is a square number or not.

Hint: *Some square numbers are: 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, etc*

Example 1:

```
Please enter positive integer: -9
Accept positive integer only!
```

Example 2:

```
Please enter positive integer: 1000
1000 is not a square number
```

Example 3:

```
Please enter positive integer: 3600
3600 is a square number
```

13. Follows these steps to create 2 functions sum and average:

Step 1: Create header file "Ex13Lib.h" as below:

```
#ifndef MYLIB13_H_INCLUDED
#define MYLIB13_H_INCLUDED

float sum(float a, float b, float c);
float average(float a, float b, float c);

#endif // MYLIB13_H_INCLUDED
```

Step 2: Create source file "Ex13Lib.cpp" as below:

```
float sum(float a, float b, float c)     { //Your code here }
float average(float a, float b, float c) { return sum(a, b, c) / 3.0; }
```

Using your library Ex13Lib to calculate rating of a pupil which based on his/her GPA (Grade Point Average) of 3 subjects literature, math and English.

Note: *the mark must be from 0.0 to 10.0.*

Mark	0.0 - 1.9	2.0 - 3.9	4.0 - 5.9	6.0 - 7.9	8.0 - 8.9	9.0 - 10.0
Rating	Poor	Fair	Average	Good	Excellent	Outstanding

Example 1:

```
Please enter mark of literature: -8
Mark of subject must be from 0.0 to 10.0!
```

Example 2:

```
Please enter mark of literature: 15
Mark of subject must be from 0.0 to 10.0!
```

Example 3:

```
Please enter mark of literature: 7.2
Please enter mark of math      : 9.0
Please enter mark of English   : 8.4
The Grade Point Average is 8.2
The rating is "Excellent"
```

14. *(*)* Follows these steps to create 2 functions Min and Max:

Step 1: Create header file "Ex14Lib.h" as below:

```
#ifndef MYLIB14_H_INCLUDED
#define MYLIB14_H_INCLUDED

float Min(float a, float b, float c);
float Max(float a, float b, float c);

#endif // MYLIB14_H_INCLUDED
```

Step 2: Create source file "Ex14Lib.cpp" as below:

```
float Min(float a, float b, float c) { //Your code here }
float Max(float a, float b, float c) { //Your code here }
```

The class has 03 pupils includes Leonardo, Remi and Ken. Using your library Ex14Lib to reward pupils based on their GPA.

Note: *the mark must be from 0 to 10.*

Example 1:

```
Please enter GPA of Leonardo: -9.2
The GPA of pupil must be from 0.0 to 10.0!
```

Example 2:

```
Please enter GPA of Leonardo: 17.5
The GPA of pupil must be from 0.0 to 10.0!
```

Example 3:

```
Please enter GPA of Leonardo: 8.5
Please enter GPA of Remi    : 9.2
Please enter GPA of Ken     : 7.9
The 1st prize is Remi
The 2nd prize is Leonardo
The 3rd prize is Ken
```

15. *(*)* Super Mind is a guessing game. In that game, player guesses the secret number and gets a reward for guessing correctly.

The game rules:

- On the first play, the player has an initial budget **M** is $50 *(M will be set to 25)*.
- Every time player starts a game, the player must pay $25 *(M will be decrease 25)*.
- The game system will generate a random secret number **V** *($1 \le V \le 100$)*.
- The player has **5** turns of guessing **T** *(T will be set to 5)*.
- The player chooses a number **N** *($1 \le N \le 100$)*. These cases occur when players guessing:
  - If the turn is over, the player loses and the system performs the following steps:
    - ✓ Displays "Game Over" message.
    - ✓ If the player still has money.
      - The system displays a message "Do you want to play again (y / n)?".
        - If 'y' is selected, the system allows the player to play the game again.
        - If 'n' is selected, the system stops and displays a goodbye message "Your money is $**M**. Thank for playing our game. See you again!".
    - ✓ If the player runs out of money.
      - The system displays a message "You runs out of money!".
      - The system stops and displays a goodbye message "Thank for playing our game. See you again!".
  - If N is equal to V, player is the winner, the prize is $50 *(M will be increase 50)*.
    - ✓ The system displays a message "Do you want to play again (y / n)?".
      - If 'y' is selected, the system allows the player to play the game again.
      - If 'n' is selected, the system stops and displays a goodbye message "Your money is $**M**. Thank for playing our game. See you again!".
  - If N is less then V, *T will be decrease 1 turn*
    - ✓ The system displays a message "Less than lucky number! Please try again.".
  - If N is greater than V, *T will be decrease 1 turn*
    - ✓ The system displays a message "Greater than lucky number! Please try again.".