**I have 8 tasks and after these tasks (below) there are requirements for this project. you can find it in the search for Rules (Project Requirements).**

**(and you will also find examples of what the code should look like there).**

**working with the maths library and arithmetic expressions. Linux programming.**

**What needs to be done?**

**Task 1:** Lift a Car

Even small children know that if they want to swing on a swing, the heavier children should sit closer to the middle. We know from physics that if the children are in balance, then the law applies:

F1 \* r1 = F2 \* r2

Archimedes wrote about the existence of levers of force: Give me a fulcrum and I will turn the whole world upside down!

Let's consider the possibility of realizing this experiment in real conditions. Let's assume that we have an iron bar with a length of 2m. In this case, where should we place the fulcrum in order to lift the car by the weight of our own body alone?

r1 + r2 = 2 => r1 = 2 – r2  
m1 \* r1 = m2 \* r2  
...  
r2 = 2 \* m1 / (m1 + m2)

Create a function float lift\_a\_car(const int stick\_length, const int human\_weight, const int car\_weight) with three parameters:

const int stick\_length - Length of the bar

const int human\_weight - Weight of a person

const int car\_weight - Car weight

According to the parameters, the function will calculate where to place the fulcrum so that a person could lift the car with his own weight. The function should return the value with the accuracy up to 2 digits after the decimal point (rounded).

### **Function Call Example**

printf("%.4f\n", lift\_a\_car(2, 80, 1400));  
// prints: 0.1100

printf("%.4f\n", lift\_a\_car(4, 90, 1650));  
// prints: 0.2100

**Task 2:** Unit Price for Toilet Paper

When shopping, price is often the deciding factor in choosing a product. Retailers should put a price per kilogram or price per quantity on every product so that shoppers can compare the price of packages of different sizes. By noticing these price tags, one quickly realizes that they are not always useful. In one case, one seller, for example, marked the price of a product per liter, another for the same product, for example, per kilogram. The third one calculated the price per piece at all, given that the goods are of different sizes in themselves.

Let's suggest a better measure of, say, toilet paper. Some packs have more rolls, some sell literally one roll at a time. Some rolls have the number of "squares" written on them, others have the total number of meters on the roll.

Let's choose one square as the measure. Since the price for one square is very small, we will calculate the price for 100 squares. Now we are left with the task of how to convert paper into squares. Simple arithmetic - 10 squares are approximately 1.17 meters.

Create a function float unit\_price(const float pack\_price, const int rolls\_count, const int pieces\_count) with three parameters:

const float pack\_price - Pack price

const int rolls\_count - Number of rolls

const int pieces\_count - Number of squares in a roll

Based on the parameters, the function calculates the price of 100 squares. The function returns the rounded result to 2 decimal digits.

### Function Call Example

printf("%.4f\n", unit\_price(4.79, 16, 150));  
// prints: 0.2000  
printf("%.4f\n", unit\_price(5.63, 20, 200));  
// prints: 0.1400

**Task 3:** ATM

An ATM dispenses banknotes in denominations of €10, €20, €50, €100 and €200. In order to withdraw money from the ATM, you need to know the number of banknotes the ATM will dispense. However, this function has not yet been programmed into the software of your ATM.

Your task is to program the function int bank\_notes(const int price) with one parameter:

const int price - Required withdrawal amount

The function calculates the number of banknotes needed for the specified amount parameter. The function returns the number of banknotes needed for withdrawal, or returns -1 if the requested amount cannot be withdrawn.

### Function Call Example

printf("%d\n", bank\_notes(540));  
// prints: 5  
printf("%d\n", bank\_notes(5));  
// prints: -1

**Task 4:** Euler's Totient Function

Euler's function, denoted as φ (fi), is a mathematical function that computes positive integers less than or equal to a given positive number n that are relatively prime to n. Two numbers are relatively prime if their greatest common divisor is 1. In other words, the Euler function φ (n) counts the number of positive integers less than or equal to n that have no common divisors (other than 1) with the number n.

Formula for the Euler function:

φ(n) = n \* (1 - 1/p1) \* (1 - 1/p2) \* ... \* (1 - 1/pk)

Where:

φ(n) is the value of the Euler function for n.

n is a positive integer.

p1, p2, ..., pk are various simple coefficients to n.

Program the function int euler(cont int n) to determine the value of the Euler function for n.

The function has one parameter:

const int n - Positive integer

### Function Call Example

printf("%d\n", euler(6));  
// prints: 2  
printf("%d\n", euler(12));  
// prints: 4

**Task 5:** Missing Number

An array contains n different numbers from the set 0, 1, 2, ..., n. The numbers form a sequence that may be in the wrong order.

Only one number is missing in this field. Your task is to use the int find\_missing\_number(const int n, const int arr[]) function to find the missing number.

The function receives 2 parameters as input:

const int n - Number of array elements.

const int arr[] - Array containing sequence values.

The function returns the value of the missing number in the array sequence.

### Function Call Example

int input\_array[] = {3,0,1};  
printf("%d\n", find\_missing\_number(3,input\_array));  
// prints: 2

**Task 6:** Pascal's Triangle

The picture shows part of Pascal's Triangle. This triangle is famous in mathematics because of its symmetry and various hidden connections. Blaise Pascal thought the same way back in 1653 and wrote that it would not be enough for him to write a scientific paper to describe it completely. The sheer number of connections of Pascal's Triangle to other areas in mathematics is what made it such a famous mathematical object.

Pascal's diagram works from top to bottom. We start with a unit (row number 0), under it we write a unit each on the right and left (row number 1). Subsequent rows are built in such a way that we write units on the sides, and the numbers in the middle are calculated as the sum of the two numbers in the previous row directly above the number to be calculated.

Your task is to calculate the sum of powers of all coefficients on a certain row, taking into account that the first row has index 0.



Create an unsigned long sum\_squared(const int line) function with a parameter:

const int line - A positive integer representing the number of a row in Pascal's Triangle

The function returns the sum of squares of all coefficients in a certain row.

### Function Call Example

printf("%lu\n", sum\_squared(1));  
// prints: 2  
printf("%lu\n", sum\_squared(4));  
// prints: 70  
printf("%lu\n", sum\_squared(33));  
// prints: 7219428434016265740

**Task 7:** Min-and-Max Price

Denis wants to make money and has a fairly simple idea - he will sell things. Since he needs revenue, he should buy at the lowest price and sell at the highest price.

Your task is to find the lowest and the highest price in the given array.

**Task 7.1**: Min

Create a function int array\_min(const int input\_array[], const int array\_size) with two parameters:

const int input\_array[] - The input is an array of integers

const int array\_size - The array value describing the number of elements in the array.

The function returns the smallest value that is in the array. If the input array is NULL, the function should return -1.

**Task 7.2**: Max

Create a function int array\_max(const int input\_array[], const int array\_size) with two parameters:

const int input\_array[] - The input is an array of integers

const int array\_size - The array value describing the number of elements in the array.

The function returns the largest value that is in the array. If the input array is NULL, the function should return -1.

### Function Call Example

int input\_array[] = {1,2,3,4,5};  
printf("%d\n", array\_min(input\_array, 5));  
// prints: 1  
printf("%d\n", array\_max(input\_array, 5));  
// prints: 5  
printf("%d\n", array\_max(NULL, 5));  
// prints: -1

**Task 8:** Factor Counter

The number of prime factors (prime factors) depends on the number itself and its decomposition into prime factors. In general, every composite number has a single prime factorization. The number of prime factors of a given number will be equal to the number of unique prime numbers that make it up.

To get the number of unique prime factors of a given number, we need to find its decomposition into prime factors. For example, having the number 12, its decomposition into prime factors is 2 and 3.Here we see that 12 has 2 distinct prime factors: 2, 2 and 3. That is, the number of different factors is 2.

Program the function int factorize\_count(const int n) to determine the number of different factors.

The function has one parameter:

const int n - Input number for decomposition.

### Function Call Example

printf("%d\n", factorize\_count(12));  
// prints: 2

Rules for the project (conditions)

**never use Ai (artificial intelligence)!!!!**

**without copying other git hubs and so on!!!!**

**Use of a global variable!!!!**

**make all these tasks in one file: arrays.c**

**If these requirements are not fulfilled, the project is not suitable for me.**

**Example of how the code should look like**

code example "guess the number game or programme"



