Problem Set 2: Numbers, Arrays

Goals

1. Practice working with a mathematical library and arithmetic expressions.
2. Understand the representation of numbers in computer memory.
3. Create custom features as specified.
4. Learn to terminate functions using different return values at different input parameters.

Task 1: Elevator and Car

Even small children know that if they want to swing on a swing, a heavier child must sit closer to the center. We know from physics that if children are to be in balance on a swing, the following must apply:

```
F1 * r1 = F2 * r2
```

Archimedes also realized the power of the lever and declared: *Give me a fixed point in space and move the Earth* .

Let's look at carrying out a similar experiment in real conditions. Suppose we have an iron rod 2 m long. Where should we place that fixed point if we just wanted to lift a car by lifting our bodies?

```
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`- Rod length
- `const int human_weight`- Human weight
- `const int car_weight`- Car weight

The function for the entered parameters calculates at which point it is necessary to support the rod so that one only lifts the car by the weight of one's own body. The function **returns a** value with the result rounded to **2 decimal places** .

Example of using the function

```
printf("%.4f\n", lift_a_car(2, 80, 1400));
// prints: 0.1100
printf("%.4f\n", lift_a_car(4, 90, 1650));
// prints: 0.2100
```

Úloha 2: Unit Price for Toilet Paper

When shopping, we often decide on the basis of price. Traders should indicate a unit price for each product so that we can compare the benefits of different packages. If we start noticing these unit prices, we will soon find that they are sometimes difficult to use. Somewhere the merchant converted the price to liters and elsewhere to kilograms. Somewhere he also converted the price to the same units, e.g. pieces, but each piece has a different size.

We will design a suitable unit and method of conversion for toilet paper. Some packages contain multiple rolls of paper, elsewhere only one roll is sold. For some reels, the number of pieces is given, elsewhere the total length in meters.

As a unit we can consider 1 fragment. Since the price of the snippet would be too low, consider the price for 100 snippets. We have to figure out how to convert meters to snippets. By a simple measurement, we find that the 10 fragments are about 1.17 meters long.

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`- Price of the package
- `const int rolls_count`- Number of discs
- `const int pieces_count`- Number of pieces in the roll

The function calculates the unit price for the entered parameters **for 100 pieces of pieces** . The function **returns a** value with the result rounded to **2 decimal places** .

Example of using the function

```
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: Collatz Conjecture

Let's take any positive integer `n`. if it is `n`an even number, we divide it by two, so we get `n/2`. if it is `n`odd number, multiply it by three, add the number `1`and we get `3n+1`. We repeat this process indefinitely.

Collatz's problem is a mathematical statement made by *Lothar Collatz* . The claim is that it doesn't matter what the starting number is `n`is chosen - the resulting sequence always comes to a number in the end `1`. For example, for the number `n=20`the sequence of numbers is as follows:

```
20, 10, 5, 16, 8, 4, 2, 1
```

Create a function `int collatz(const int number)`with parameter:

- `const int number`- Positive nonzero integer (is the beginning of the sequence)

The function **returns the** length of the sequence for the input number `n`. In the example shown for number `20`will be the result `8`.

Example of using the function

```
printf("%d\n", collatz(20));
// prints: 8
printf("%d\n", collatz(35));
// prints: 14
```

Task 4: Find Opposite Number

Consider integers from `0` do `n - 1`which are written along the circle. It is true that the distance between two adjacent numbers is the same.

Create a function `int opposite_number(const int n, const int number)`with two parameters:

- `const int n`- Number of numbers along the circle, positive nonzero even number
- `const int number`- A positive integer less than `n`

The function **returns the** value of the number that is opposite the number `number`.

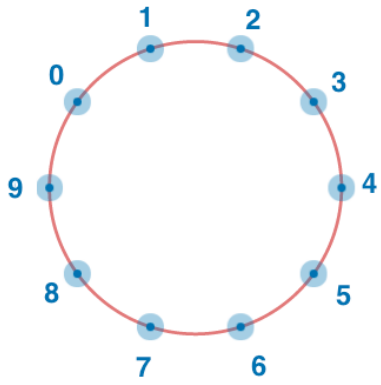

FIG. 1: Illustrative image

Example of using the function

```
printf("%d\n", opposite_number(10, 2));
// prints: 7
printf("%d\n", opposite_number(12, 9));
// prints: 3
```

Úloha 5: Count Numbers in Array

A group of people stand in line and wait to be included in the teams, where:

- 1. the person belongs to the 1st team
- 2. the person belongs to the 2nd team
- 3. the person belongs to the 1st team
- ...

Each person is represented by a positive integer. Your task is to find out the overall rating of the 1st team and the overall rating of the 2nd team.

Create a function `void counter(const int input_array[], const int array_size, int result_array[2])`with three parameters:

- `const int input_array[]`- Input field of positive integers
- `const int array_size`- The size of the input field, expressed by the number of elements in the field
- `int result_array[2]`- Output field, its size will always be 2 elements

The function **does return** not any value.

The function calculates and **writes the results to the field** `result_array[]`according to the following rules:

- Element `result_array[0]`will contain the sum of all the elements that are in the field `input_array[]`in even positions
- Element `result_array[1]`will contain the sum of all the elements that are in the field `input_array[]`in odd positions

> **Note :** In this role `0`considers an even number.

Example of using the function

```
int input_array[] = {1,2,3,4,5};
int result_array[2];
counter(input_array, 5, result_array);
printf("%d %d\n", result_array[0], result_array[1]);
// prints: 9 6
```

Problem 6: Pascal's Triangle

The picture shows a part of Pascal's triangle. Pascal's triangle became famous in mathematics due to its symmetry and various hidden relationships. Blaise Pascal thought the same thing in 1653 and noted that he probably could not describe them in one work. The many connections between Pascal's triangle and other branches of mathematics have made it a sacred mathematical object.

Pascal's scheme works from above. Let's start with one unit (line no. `0`) and under it we write two units left and right (line no. `1`). We construct the next lines by writing 1 on both edges and we get the middle numbers as the sum of a pair of numbers in the previous line directly above where we are.

Your task is to calculate the sum of the powers of all the coefficients in the selected row, if the rows are numbered from `0`.

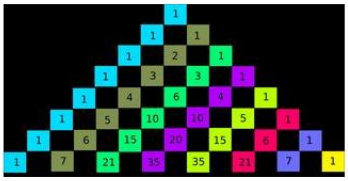

FIG. 2: Pascal's triangle

Create a function `unsigned long sum_squared(const int line)`with parameter:

- `const int line`- A positive integer that represents a line in Pascal's triangle

The function **returns the** sum of the squares of all the coefficients in the given line of the Pascal triangle.

Example of using the function

```
printf("%lu\n", sum_squared(1));
// prints: 2
printf("%lu\n", sum_squared(4));
// prints: 70
printf("%lu\n", sum_squared(33));
// prints: 72194284340162657 40
```

Task 7: Min-and-Max Price

Denis wants to make money and got a very simple idea - he will sell things. Because he wants to make some profit, he needs to buy things at the lowest possible price and sell at the highest possible price.

Your task is to find the lowest and highest price in the box that is in the input.

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[]`- Input field containing integers
- `const int array_size`- The size of the input field, expressed by the number of elements in the field

The function **returns the smallest value** that is between the elements of the input field. However, if the input field is `NULL`, the function returns a value `-1`.

Task 7.1: Max

Create a function `int array_max(const int input_array[], const int array_size)`with two parameters:

- `const int input_array[]`- Input field containing integers
- `const int array_size`- The size of the input field, expressed by the number of elements in the field

The function **returns the largest value** that is between the elements of the input field. However, if the input field is `NULL`, the function returns a value `-1`.

Example of using functions

```c
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: Special Counter

Program the function `unsigned long special_counter(const int input_array[], const int array_size)`with two parameters:

- `const int input_array[]`- Input field containing integers
- `const int array_size`- The size of the input field, expressed by the number of elements in the field

The function **returns the** sum of all elements of the array. Numbers that are in in the field **odd** positions need be **to amplified** .

> **Note :** In this role `0`considers an even number.

Example of using the function

```c
int input_array[] = {11,12,13,14,15};
printf("%lu\n", special_counter(input_array, 5));
// prints: 379
```

```
--> 11 + 13 + 15 + 12^2 + 14^2
= 11 + 13 + 15 + 144 + 196
= 379
```

Task 9: Special Numbers

Program the function `int special_numbers(const int input_array[], const int array_size, int result_array[])`with three parameters:

- `const int input_array[]`- Input field of positive integers
- `const int array_size`- The size of the input and output field, expressed by the number of elements in the field
- `int result_array[]`- Output field, its maximum size is expressed by a parameter `array_size`

The function **returns the** number of all special numbers found.

All special numbers found must be **entered in the field** `result_array[]`. A number is special if it is greater than the sum of all numbers that have a larger index in the field than the number (which are to the right of it in the field).

Example of using the function

```
int input_array[] = {16,17,4,3,5,2};
int result_array[6];
int count = special_numbers(input_array, 6, result_array);
for(int i = 0; i < count; i++){
    printf("%d ", result_array[i]);
}
printf("\n");
// prints: 17 5 2
```