IN8011 - Additional Exercises

Moritz Makowski - moritz.makowski@tum.de

This is just a collection of some exercises.

Please try to solve them before looking at the solutions. There are many possible solutions to a given problem. The given solutions are just the ones I came up with.

Exercise 1: Average Lifetime of Bacteria ★ ☆ ☆ ☆ ☆

Write a simulation to determine the average lifetime of individual bacteria organisms. Each individual organism dies with a probability p after one timestep.

You may use the following lines to generate a random number.

```c
#include <stdlib.h>
#include <time.h>

int main() {
    srand(time(NULL));
    int my_random = rand() % 42;  // A random integer between 0 and 42
}
```

Exercise 2: Rock-Paper-Scissors ★ ★ ☆ ☆ ☆

Write a program that plays Rock-Paper-Scissors against a human player.

You may use the following lines to generate a random number.

```c
#include <stdlib.h>
#include <time.h>

int main() {
    srand(time(NULL));
    int my_random = rand() % 42;  // A random integer between 0 and 42
}
```

Exercise 3: String Palindrome ★ ★ ☆ ☆ ☆

Check whether a given **string** is a palindrome.

```
char str1[10] = "abcba";
char str2[10] = "abcbaaa";
is_str_palindrome(str1);  // returns 1
is_str_palindrome(str2);  // returns 0
```

Exercise 4: String Anagram ★ ★ ★ ☆ ☆

Check whether two given **strings** are an anagram.

```
char str1[10] = "listen";
char str2[10] = "silent";
char str3[10] = "hear";
is_str_anagram(str1, str2);  // returns 1
is_str_anagram(str2, str3);  // returns 0
```

You can add case-insensitivity in there if you want ;)

Exercise 5: Integer Palindrome ★ ★ ★ ★ ☆

Check whether a given **integer** is a number-palindrome.

```
int num1 = 4218124;
int num2 = 4124;
is_int_palindrome(num1);  // returns 1
is_int_palindrome(num2);  // returns 0
```

Exercise 6: Integer Anagram ★ ★ ★ ★ ★

Check whether two given **integers** are an anagram.

```
int num1 = 30017;
int num2 = 10730;
int num3 = 30010;
is_num_anagram(num1, num2);  // returns 1
is_num_anagram(num2, num3);  // returns 0
```

Bonus: Leading zeros can be added to achieve an anamgram-relation:

```
int num1 = 30017;
int num2 = 317;
is_num_anagram(num1, num2);  // returns 1
```

Exercise 7: Parentheses Logic ★★☆☆☆ (★★★★☆)

Write a program that tests whether a given string including parentheses (`(` and `)`) is fulfilling the rules for setting parentheses:

1. For every opening parentheses there exists a closing parentheses and vice versa
2. Every closing paretheses appears after the respective opening parentheses.
3. Other characters do not play a role in this logic

Super Bonus: Expand your program to support different types of braces -> `(` / `)`, `{` / `}`, `[` / `]`. It is important that they don't interfere with each other, e.g. `([...)]` is invalid!

Exercise 8: Calculate the Checksum (Quersumme) ★ ☆ ☆ ☆ ☆

Write a program that calculates the checksum of a decimal number.

The checksum of a number is the sum of all digits of the number.
E.g. The checksum of `2345` is `14` because `2+3+4+5 = 14` .

Exercise 9: Calculate Harshad-Numbers ★★☆☆☆

An integer number is called harshad number if it is evenly divisible by its checksum.

Write a program that calculates the first 100 harshad-numbers.

Exercise 10: Calculate Perfect Numbers ★ ★ ☆ ☆ ☆

An integer number is called perfect number if it is equal to the sum of its even divisors:

The first two perfect numbers are:

- 6 = 3 + 2 + 1
- 28 = 14 + 7 + 4 + 2 + 1

Write a program that calculates the first 4 perfect numbers.

Exercise 11: Equation-Strings ★ ★ ★ ☆ ☆ (★ ★ ★ ★ ☆)

Write a program in which you defined a String inside the code which contains a mathematical expression and solve that expression. The string can only contain digits, `+` and `-` signs. No spaces or other characters.

Example:

```
char math_string[100] = "1+40-55-30+16";

int result = solve_equation(math_string, 100);  // result = -28
```

Bonus Idea: Whenever an equal sign `=` is found inside a string the thing is treated as an equation.

Example:

```
char equation_1[100] = "3-4+15=14";
char equation_2[100] = "3-4+15=17";
char equation_3[100] = "3-4+15=14=16-2";

int result = solve_equation(equation_1, 100);  // returns 1 (true)
int result = solve_equation(equation_2, 100);  // returns 0 (false)
int result = solve_equation(equation_3, 100);  // returns 1 (true)
```

Exercise 12: Equation Possibilities ★ ★ ★ ★ ☆

Write a program that outputs all possibilities to put `+` or `-` or *nothing* between the numbers 1, 2, ..., 9 (in this order) such that the result is 100.

For example 1 + 2 + 3 - 4 + 5 + 6 + 78 + 9 = 100.

Exercise 13: Conway's Game of Life ★ ★ ★ ★ ☆

Write conways game of life in C. Here is an article explaining the simulation:
https://en.wikipedia.org/wiki/Conway's_Game_of_Life

1. Generate a random matrix
2. Draw (print) the matrix to the console
3. Evolve the matrix by one simulation step
4. Write an infinite loop (can be stopped with CTRL+C) for steps 2 and 3

The result should look something like this:
https://www.youtube.com/watch?v=p4xh7-Mdmlc or this
https://www.youtube.com/watch?v=dBS99UWrTmQ

What about the edge of the matrix? You can try three different strategies:

- "Activating edge" = All Edge cells are "alive"
- "Deactivating edge" = All Edge cells are "dead"
- Infinite Plane = The Square is repeated indefinitely -> Indices ... 3 4 **0 1 2 3 4** 0 1 ...
- Mirror -> 1 **0 1 2 3 4** 3

I would go for an infinite plane but you could also try out any other approach ;)

Sources (including more challenges)

- https://adriann.github.io/programming_problems.html
- https://projecteuler.net/