Additional Exercises

Group 06 - Moritz Makowski

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
}
```

Solution: `exercise_1_average_lifetime_of_bacteria.c`

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
}
```

Solution: `exercise_2_rock_paper_scissors.c`

Exercise 3: 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

Solution: `exercise_3_parentheses_logic.c`

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!

Solution: `exercise_3_advanced_parentheses_logic.c`

Exercise 4: 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 $2 + 3 + 4 + 5 = 14$.

Solution: `exercise_4_calculate_checksum.c`

Exercise 5: 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.

Solution: `exercise_5_harshad_numbers.c`

Exercise 6: 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.

Solution: `exercise_6_perfect_numbers.c`

Exercise 7: Equation-Strings ★ ★ ★ ☆ ☆ (★ ★ ★ ★ ★)

Write a program in which you defined a String inside the code which contains a mathematical expression and solve that expression.

Example:

```
char math_string[100] = "3*4+15+78-3*5";

// solve_equation is the function that you should write
// Its signature isn't required to look like this.
int result = solve_equation(math_string, 100);  // returns 90
```

You can set your desired **level of difficulty** and work your way up:

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

char medium_string_1[100] = "1+40*60-55-30/16";
char medium_string_2[100] = "1+40*(55-30)+16";

char hard_string[100] = "1+40*[(55-30)*4+16]";
//   should be equal to "1+40*((55-30)*4+16)"
//                   or "1+40*([55-30]*4+16)"
```

Solutions: `exercise_7_equation_strings/`

Additional 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=27";
char equation_2[100] = "3*4+15=23";
char equation_3[100] = "3*4+15=4*5+7=5*5+2";

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

Solution *will follow shortly.*

Exercise 8: 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.

Solution *will follow shortly.*

More Exercises

Coming soon!

All **code examples** and **exercise solutions** on **GitHub**.

https://github.com/dostuffthatmatters/Engineering-Informatics-1-MSE-WS1920.

Resources

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