Introduction to Program Synthesis - Exercise I (Block A)
S-Expression and Syntax tree

Deadline: 29-11-2024 (Anywhere on Earth)

Through the following exercises you will set up basic structures which will be used in future exercises. Do your best to keep your code clean and modular.

We are working with S-expressions. Please refer to your course material for more details about their syntax.

Exercise 1: Expression tree

Consider the following mathematical expression:

$$(x - (\frac{y}{4})) + (z * (t^2))$$

Draw the above s-expression as an expression tree and write down a proper textual representation that can be used for parsing the expression..

Exercise 2: Syntax tree

Consider the following expression:

```
defun foobar (
   (setq a 1)
   (setq b 3)
   (if
     (< a b)
     (setq c (- b a))
     (setq c (- a b))
   )
 )
```

For the given example, draw the corresponding syntax tree.

Exercise 3: Tree implementation and parsing

Based on the C parser seen during the lecture, write a parser in Python to read a textual representation of the above expression in Exercise 1 and store it in a syntax tree.
Traverse and print the tree obtained and compare them to your previous answers to check for correctness.
For this exercise the following tree function should be implemented:

- `create_tree`
- `create_node`
- `remove_tree`
- `traverse`

> **Hint:** In addition to the code example from the lecture, you can generally orientate on the the example of a binary search tree. It an also help to sketch the dependencies of the respective nodes and edges on a sheet of paper.