GP 2	**Generative Programming**	**ST 17, Exercise 2**

Deadline: 21.04.2017, 13:50

- ☐ **Gr. 1,** **J. Karder, MSc.**
- ☐ **Gr. 2,** **P. Fleck, MSc.**

Name ________________________________ **Effort in h** _____

Points ________________________________ **Lecturer** ______

1. JSON Validator (10 Points)

JSON (JavaScript Object Notation) is a popular format for data exchange between servers and (web) applications.

Implement a simple parser with Boost Spirit that is able to read a JSON file and verify its syntactical correctness. It should at least be able to parse the following snippet:

```
{
    "Image": {
        "Width": 800,
        "Height": 600,
        "Title": "View from 15th Floor",
        "Opacity": 0.5,
        "Thumbnail": {
            "Url": "http://www.example.com/image/481989943",
            "Height": 125,
            "Width": 100
        },
        "Location": {
            "Latitude": 37.7668,
            "Longitude": -122.3959
        },
        "Animated": false,
        "IDs": [116, 943, 234, 38793]
    }
}
```

Please note that the parser should be able to handle objects, members, pairs, arrays, elements and values. You do not need to take care of special characters as well as special number formats like the E notation.

You can find the JSON specification at http://json.org/ and the RFC at http://tools.ietf.org/html/rfc7159.

2. Mini-LOLCODE with Boost.Spirit (4 + 1 + 2 +3 + 2 + 2 Points)

LOLCODE is an esoteric programming language with a syntax resembling lolspeak (the language of the lolcats), e.g.:

```
HAI
CAN HAS STDIO?
VISIBLE "HAI WORLD!"
KTHXBYE
```

Implement a simple interpreter for LOLCODE using Boost Spirit. It should be able to parse LOLCODE source code, execute the code and print to the console:

- First, implement an interpreter that can handle the above example. The `VISIBLE` statement can take any expression that returns a value and prints it to the console. Note that by default a line break is added to the output. To permit line breaks, VISIBLE can be terminated with an exclamation mark. `CAN HAS` is used to include libraries. The parser should be able to handle includes, but no further action has to be performed.

- Single line comments in LOLCODE start with `BTW` and end at the end of a line.

- Multi line comments start with `OBTW` and end with `TLDR`.

- Variables are declared with the `I HAS A` statement followed by the name of the variable. To assign a value to a variable, the `R` keyword is used:

  ```
  I HAS A variable
  variable R 5.0
  ```

  LOLCODE automatically detects the data type of the variable. Implement at least the data type double (`NUMBAR`) and bool (`TROOF`).

- Implement at least 4 basic arithmetic operations, e.g.:

  ```
  SUM OF 4 AN 5
  ```

  It's sufficient for the interpreter to be able to handle numbers and nested mathematical operations. The result of an arithmetic operation can be assigned with `R` to a variable. If there is no assignment, it is saved in a temporary variable called `IT`.

- Implement at least 4 Boolean operations, e.g.:

  ```
  BOTH OF WIN AN FAIL
  ```

  The result of a Boolean operation can be again assigned to a variable or saved in `IT` if omitted.

For more information about the LOLCODE syntax you can have a look at the specification at https://github.com/justinmeza/lolcode-spec/blob/master/v1.2/lolcode-spec-v1.2.md.