Andrea Tesei
Matr: 464561

Exercise 1

To develop the “FunW@P” project i’ve choosed the “FsLexYacc” toolbox, which is a project that contain a Lexer generator (FsLex.exe) and a Parser generator (FsYacc.exe) that produce the outputs in F# code.
This toolbox produces a lexer for byte and Unicode character input and a LALR(1) parser: this type of Look-Ahead Left-to-Right parser is a subtype of the general and canonical Left-to-Right parser (LR) which is a Bottom-up parser. The 'L' means that this type of parsers reads the input stream from left to right, and the 'R' means that the parser chooses the rightmost derivation.
It uses a static transition table to parse the given token stream, which is produced by the lexer: the parser uses the lookahead operation to choose the right derivation.
FsLexYacc accept context-free grammar and give to the programmer the ability of produce the elements of the abstract syntax tree directly: this is the first reason for which i choose this program as Lexer/Parser generator, and the other one is that it generates F# code, which is more powerful for developing this type of program.
The files in the project are:

- **Program.fs**: contains only the EntryPoint of the program with the main function which take as argument the path of the file of the source code;

- **Ast.fs**: this file contains the definition in F# of the type nodes of the abstract syntax tree;

- **Lexer.fsl** and **Lexer.fs**: the former is the definition of the rules for FsLex.exe, which are used to define the tokens of the language and the regular expression to parse the source code of this language; the latter is the file generated by the program FsLex.exe;

- **Parser.fsp, Parser.fsi** and **Parser.fs**: the first is the file in which the FunW@P’s context-free grammar, which is defined in such a

way that deal with the syntax of the FsYacc program; the second and the third are the files generated by FsYacc;

- **WebServiceCli.fs**: is the file in which is implemented the Web Service client used by the “dasync” statement;

- **Environment.fs**: in this file the environment for the interpreter and the memory of the global function definitions are implemented;

- **Interpreter.fs**: for this language i’ve choose to develop only the interpreter which is implemented in this file as a set of recursive functions.

To run this program, symply compile it in this order: Ast.fs, Parser.fsi, Parser.fs, Lexer.fs, Environment.fs, WebServiceCli.fs, Interpreter.fs and Program.fs. Then run in the path of the output:

FunW_P.exe “path-to-source-code”

Project choices:

- the Dasync is able only to manage functions which their definition is global: it doesn't accept anonymous function, which are implemented in this language. It is also able only to accept function that return Integer or Boolean types.

- The environment is implemented as a List of Maps of “eval”, to exploit the concept of stack: there is also the concept of closure, to implement the anonymous function feature.

- The type checking phase is done in the interpreter at run-time.

- To serialize/deserialize object in json i use the Newtonsoft Json.NET framework.

Exercise 2

The second project is called “SHWebService” and implements the back-end of the remote server in which the Dasync calls are executed. For this project i have used both C# and F# code: the latter is utilized in the FunW@PLib in which there is the implementation of the function which use the FunW@P function to execute the functon call; the former is used to implement the Web server and the Web service Dasync.
The files of the projects are:

- **Program.cs**: in this file the front-end and the Web service definition are implemented, and this contains also the entrypoint of the program;

- **DeserializeDasyncRequest.cs**: this file contains the definition of the functions used to deserialize the DasyncRequest received by the Web service;

- **Library1.fs**: in this file the real Dasync function is implemented exploiting the functions of the FunW@P project. This is the only file of the FunW@PLib project.

Even in this project is use the Newtonsoft Json.NET framework to deserialize/serialize object from/to json and i have implemented also a set of functions to deserialize the custom types of the webservice's parameters which are received from the client of FunW@P.
To use it, simply run the program SHWebService.exe from command prompt.