Design an original programming language and provide a syntactic analyzer for it, using YACC. Your language should should include

- Predefined types: int , float, char, string, bool (you can use your own syntax for the values of the char, string, and bool types)
- user defined data types (similar to classes in object orientated languages, but with your own syntax); provide specific syntax to allow initialization and use of variables of these types
- array types
- variable declarations/definition, constant definitions
- control statements (if, for, while, etc.), assignment statements
- arithmetic and boolean expressions
- operations with string types
- function calls which can have as parameters: expressions, other function calls, identifiers, constants,etc.
- A predefined function called *Eval* which has a parameter of typr int

Your yacc analyzer should:

1) do the syntactic analysis of the program

2) create a symbol table for every input source programm in your language, which should include:

- information regarding variable or constant identifiers (type, value – for identifiers having a predefined type, scope: global definition,defined inside a function , defined inside a user defined type, or any other situation specific to your language)
- information regarding function identifiers (the function signature, whether the function is a method in a class etc)

The symbol table should be printable in a file (symbol_table.txt)

3) provide semantic analysis, checking that:

- any variable that appears in a program has been previously defined;
- a variable should not be declared more than once;
- a variable appearing in the right side of an expression should have been initialized explicitly.
- a function is not defined more than once with the same signature
- a function that is called in the program has been defined
- the left side of an assignment has the same type as the right side
- the parameters of a function call have the types from the function definition
-in any call to the Eval, the parameter has type int.

Error messages should be provided if these conditions do not hold;

4) provide the evaluation of integer arithmetic expressions in a program ; if a program is syntactically and semantically correct and the *expr* type is int , for every call of the form *Eval(expr)* , the actual value of expr will be printed.

Besides the homework presentation, students should be able to answer specific questions regarding grammars and parsing algorithms (related to the first part of your homework) or yacc details related to the second part (the answers will also be graded).