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

- variable declarations, constant declarations (constants with several types) , function declarations and definit
- array types
- user defined data types (similar to classes in object oriented languages, but with your own
- syntax); provide specific syntax for working with variables of these types
- control statements (if, for, while, etc.), assignment statements
- arithmetic and boolean expressions with complex operands
- operations with string types
- function calls which can have as parameters: expressions, other function calls, identifiers, constants,etc.
- A predefined function called *Print* which has as parameters: an expression and a constant which designates a type

For every input source file, your yacc analyzer should:

1) do the syntactic analysis

2) create a symbol table, which should include:

- information regarding variable or constant identifiers (type, value, scope: 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)
- information regarding identifiers that represent user defined type names (the scope, what kind of type it is)

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

3) provide semantic analysis - check 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

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

4) provide the evaluation of arithmetic expressions in a program ; if a program in syntactically and semantically correct, for every call of Print(*expr*, *type*) , the actual value of expr will be printed, if the *expr* has the right type specified by the *type* argument.

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).