CS 315

PROGRAMMING LANGUAGES

Project 1

Language Name: XTML--

Instructor: H. Altay Güvenir

Group 19

Section 1

Deniz Sun - 22003981

Mehmet Emre Kantaş - 22003693

Mustafa Hamit Dölek - 21703136

Part A

1. **BNF**

```
<program> ::= start <stmts> finish

<stmts> ::= <stmt>

             | <stmt>  ; <stmts>

<stmt> ::= <assign_stmt>

            | <logical_stmt>

            | <if_stmt>

            | <loop_stmt>

            | <return_stmt>

<return_stmt> ::= return <expr>;

<assign_stmt> ::= <var> <assignment_op> <expr>

<expr> ::= <expr> <logical_op> <expr>

            | <var>

            | <not_op> <expr>

<not_op> ::= ~

<logical_op> ::= && | => | ⇔| <or_op> | ==

<assignment_op> ::=  =

<or_op> ::= ||

<var> ::= <lower_char>

<true> ::= 1

<false> ::= 0

<bool> ::= <true> | <false> | 1 | 0

<logical_stmt> ::= (<var><logical_op><logical_stmt>)

                    | <var>

<if_stmt> ::= if <logical_stmt> { <stmts> }
```

```
                | if <logical_stmt> { <stmts> } else { <stmts> }

<loop_stmt> ::= while <logical_stmt> { <stmts> }

<function_name> ::= “foo” | “func1” | “func2” | “func3” | ”func4” | “func5” | ”func6” | “func7”

<function_def> ::= <function_name> (<param>) { <stmts> <return_stmt>}

                        | <function_name> () { <stmts> <return_stmt>}

                        | <function_name> () { <stmts> }

                        | <function_name> (<param>) { <stmts> }

<param> ::= <var>

                | <var>, <param>

<array_name> = “arr1”, “arr2”, “arr3”, “arr4”, “arr5”, “arr6”, “arr7”, “arr8”

<array_structure> ::= array <array_name><assignment_op> [<elems>]

                | array <array_name><assignment_op>[]

<elems> ::= <elem>

                | <elem>, <elems>

<elem> ::= <var>

                | <bool>

<comment> ::= /* <string> */

                | // < string>

<input_statement> ::= <var> <assignment_op> in();

<input_string> ::= “true” | “false” | “1” | “0” | “True” | “False” | “TRUE” | “FALSE”

<output_statement> ::= out(<expr>);

<string>::= “” | “ <char> ” | “ <char> <string> ”

<char>::= <upper_char> | <lower_char>

<upper_char>::= A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W
| X | Y | Z

<lower_char>::= a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z
```

2. Definitions of BNF

<program>: This non-terminal starts the program for this language.

start: This terminal indicates that the program has started.

finish: This terminal indicates that the program has finished.

<stmts>: This non-terminal is the statements list for implementing one or more statements.

<stmt>: This non-terminal indicates a single statement that can be either one of: assign statement, logical statement, if statement, loop statement, return statement.

<assign_stmt>: This non-terminal indicates a single statement that assigns a variable to an expression.

<logical_stmt>: This non-terminal indicates a statement can be only a variable or a sequence of a variable, logical operator and another logical statement. In either way, this statement means a boolean value.

<if_stmt>: This non-terminal indicates either a single if statement or an if-else statement.

<loop_stmt>: This non-terminal indicates a simple while-loop statement.

<return_stmt>: This non-terminal returns a value from the callee function to the caller.

<expr>: This non-terminal indicates anything that ends up being a boolean value, true or false. This non-terminal can be derived to either one of: 2 other expressions separated by a logical operator, a variable with a not operator on its left or a single variable.

<var>: This non-terminal indicates a value that can change in the execution of the program.

<assignment_op>: This non-terminal indicates the terminal ‘=’. This operator is used in assignment statements.

<logical_op>: This non-terminal can be derived to the terminals: ‘&&’, ‘->’, ‘<->’ and ‘==’, or to the non-terminal or operator.

<not_op>: This non-terminal indicates the terminal ‘~’ which is a unary operator that changes the boolean value true to false and false to true.

<or_op>: This non-terminal indicates the terminal ‘||’ which means logical or operator.

<true>: This non-terminal indicates the boolean value of true.

<false>: This non-terminal indicates the boolean value of false.

<bool>: This non-terminal value indicates either true, false, 1 or 0. (1 and 0 also means true and false, respectively.)

<function_name>: This non-terminal can be derived to the strings: “foo”, “func1”, “func2”, “func3”, ”func4”, “func5”, ”func6” or “func7”. These strings can be used as function names.

<function_def>: This non-terminal can be used to define functions. These defined functions may have parameter lists with either no parameters, one single parameter or multiple parameters. Furthermore, these defined functions may or may not have a return statement.

<param>: This non-terminal indicates a parameter list for a function. This can be derived to either a single variable or a set of variables separated by comma.

<array_name>: This non-terminal can be derived to the strings: “arr1”, “arr2”, “arr3”, “arr4”, “arr5”, “arr6”, “arr7” or “arr8”.

<array_structure>: This non-terminal indicates the array data structure. It consists of the terminal ‘array’, an array name, an assignment operator and the set of elements in square brackets, separated by comma.

<elems>: This non-terminal indicates either a single element or a set of elements, separated by comma.

<elem>: This non-terminal indicates either a variable or a boolean variable.

<comment>: This non-terminal is used for creating comments that will not affect the flow of execution of the program.

<string>: This non-terminal indicates a string value that will be used for writing comments.

<char>: This non-terminal indicates a char data type that makes up the string.

<upper_char>: This non-terminal is for upper case char values.

<lower_char>: This non-terminal is for lower case char values.

<input_statement>: This non-terminal indicates the syntax of the input function of this language. It consists of a variable on the left-hand side, an assignment operator, the terminals ‘in’, ‘(’ and ‘)’. Input function will take a boolean value from the user at execution-time and assign that value to the variable on the left-hand side.

<input_string>: This non-terminal indicates the values that the input function can take. These values are the strings: “true”, “false”, “1”, “0”, “True”, “False”, “TRUE” and “FALSE”. If the user inputs some value other than these values, the input function will ask for a new input from the user.

<output_statement>: This non-terminal indicates the syntax of the output function of this language. It consists of the terminal ‘out’, ‘(’, ‘)’ and between the parentheses, the expression that will be output.

return: This terminal is used in writing the return statements as a keyword.

~: This terminal indicates the logical not operator. This operator gives a boolean value.

&&: This terminal indicates the logical and operator. This operator gives a boolean value.

->: This terminal indicates the logical implication operator. This operator gives a boolean value.

<->: This terminal indicates the logical double-implication operator. This operator gives a boolean value.

||: This terminal indicates the logical or operator. This operator gives a boolean value.

==: This terminal indicates the relational equals operator. This operator gives a boolean value.

=: This terminal indicates assignment. It is used in the assignment statements.

if: This terminal is used in defining the if part of the if-else statements as a keyword.

else: This terminal is used in defining the else part of the if-else statements as a keyword.

while: This terminal is used in defining while-loop statements as a keyword.

array: This terminal is used in defining array data structures as a keyword.

/*: This terminal indicates the start of a comment.

*/: This terminal indicates the end of a comment.

in: This terminal is used for the input function as a keyword.

out: This terminal is used for the output function as a keyword.

(: This terminal indicates the left-parenthesis. It appears in defining functions, logical statements and using input and output functions.

): This terminal indicates the right-parenthesis. It appears in defining functions, logical statements and using input and output functions.

{: This terminal indicates the left curly parenthesis. It starts the contents of a function or statement.

}: This terminal indicates the right curly parenthesis. It ends the contents of a function or statement.

;: This terminal ends the line of statement.

,: This terminal separates variables.