2019-2020 SPRING SEMESTER

CS315 PROGRAMMING LANGUAGES

LANGUAGE NAME: TRIP

PROJECT - 1 REPORT(GROUP 26)

STUDENT NAME: BERK YILDIZ

STUDENT ID: 21502040

SECTION : 1

STUDENT NAME: BURAK BAYAR

STUDENT ID: 21401971

SECTION : 2

BNF DESCRIPTION OF TRIP PROGRAMMING LANGUAGE

<program> -> <stmts>

<stmts> -> <stmt> | <stmt>; <stmts>

<stmt> -> <if_stmt> | <non_if_stmt>

<if_stmt> -> <matched> | <unmatched>

<matched> -> if (<logic_expr>) <matched> else <unmatched> | <non_if_stmt>

<unmatched> -> if (<logic_expr>) <if_stmt>

|if (<logic_expr>) <matched> else <unmatched>

<logic_expr> -> <simple_logic_expr> | <logic_expr> <logic_op> <logic_expr> | (<logic_expr>) | !(<logic;_expr>)

<non_if_stmt> -> <call_stmt> | <input_stmt> | <assignment_stmt> | <loop_stmt> | <output_stmt> | <return_stmt> | {<stmts>}

<call_stmt> -> <func_identifier> (<element_list>) | <func_identifier>( )

<input_stmt> -> insert <var_identifier>

<assignment_stmt> -> <var_identifier> \= <var_identifier> | <var_identifier> \= <number>

<loop_stmt> -> <for_stmt> | <while_stmt>

<for_stmt> -> for(<var_identifier>: in <set>) {<stmts>}

<while_stmt> -> while(<logic_expr>) {<stmts>}

<output_stmt> -> get <set>

<return_stmt> -> return <set>;

<set> -> <types> | {<element_list>} | [<element_list>]

<element_list> -> <types> | <element_list>, <types>

<types> -> <variable_identifier> | <function_identifier> | <string> | <number>

<string> -> "<char_array>"

<variable_identifier> -> <lowcase_letter> | <lowcase_letter> <alphnumeric_string>

<function_identifier> -> <upcase_letter> | <upcase_letter> <alphanumeric_string>

<number> -> <signed_integer> . <unsigned_integer>

<lowcase_letter> -> 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

<upcase_letter> -> 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

<letter> -> <upcase_letter> | <lowcase_letter>

<sign> -> +|-

<signed_integer> -> <digit> | <sign> <signed_integer> | <digit> <signed_integer>

<unsigned_integer> -> <digit> | <digit> <unsigned_integer>

<digit> -> 0|1|2|3|4|5|6|7|8|9

<char> -> <alphanumeric> | : | ; | $ | _ | - | / | + | * | . | , | ? | ! | (|) | [|] | { | } | ^ | % | =

<char_array> -> <char> | <char_array> <char>

<alphanumeric> -> <letter> | <digit>

<alphanumeric_string> -> <alphanumeric> | <alphanumeric> <alphnumeric_string>

<logic_op> -> && | \|\|

<set_relations> -> > | < | != | ==

<simple_logic_expr> -> <set> <set_relations> <set>

<comment> -> #<char_arr>

<space> -> \t | \

Description of BNF: Trip Programming Language consists of statement(s) which are made of statements. These statements are set for preventing ambiguity. In Trip language there are two types of statements and they are if statements and non-if statements. As many languages divide if statements into two types which are matched and unmatched, we did the same for Trip Language. By doing this, we both avoided ambiguity and allowed programmers to write nested if statements. For checking the conditions, there was a need for logic expressions. Also, as mentioned, non-if statements are included in this language and those are call statements, input statements, assignment statements, loop statements, output statements and return statements. Also, in our language, primitive data types we have types and some needed types like string(also alphanumeric string), char, number, integers(signed and unsigned) digits and comments, spaces etc.

In order not to mention everything, you can find just the important parts of BNF description below.

<program> Program starts with statements.

<stmts> -> Statements consist of one or more statements.

<stmt> -> There are two types for statement, if or non-if statement.

<if_stmt> -> Matched and Unmatched are the types of if statements.

<matched> -> If statement following with matched and else following with unmatched statement or non if statement are included in that notation.

<unmatched> -> if statement and if with else statement is unmatched.

<logic_expr> -> This is the logic expression with not notation.

<non_if_stmt> -> Call statements, input statements, assignment statements, loop statements, output statements and return statements are non-if statements.

<call_stmt> -> It is for calling statement.

<input_stmt> -> Variable Identifier is needed for input statement.

<assignment_stmt> -> This is for assignment statement.

<loop_stmt> -> There are two types of loop statements which are for and while.

<for_stmt> -> For statement following with statement is included there.

<while_stmt> -> Simple While Statement is presented here.

<output_stmt> -> By writing get, output statement is called.

<return_stmt> -> Return statement following with set is the Return Statement.

<set> -> Set is for types and can be added.

<element_list> -> Element list and types are stated here.

<types> -> Variable identifier or function identifier or string or number is for types.

<string> -> String is the char array.

<variable_identifier> -> Lowercase letter and alphanumeric string are variable

<function_identifier> -> Uppercase letter, alphanumeric string are needed.

<number> -> Signed integer following dot and unsigned integer number is the number.

<lowcase_letter> -> These are lowercase letters.

<upcase_letter> -> These are uppercase letters.

<letter> -> Uppercase or lowercase are the types.

<sign> -> Used in numbers.

<signed_integer> -> Signed integers need <sign>.

<unsigned_integer> -> Unsigned integer does not require <sign>.

<digit> -> These are the digits.

<char> -> Collection of numbers, strings and special characters.

<char_array> -> Array of char.

<alphanumeric> -> Consist of letters and digits.

<alphanumeric_string> -> Consist of one or more alphanumeric.

<logic_op> -> They are needed for logic operations.

<set_relations> -> For logic comparisons, they are used.

<simple_logic_expr> -> That decides how simple logic expression should be.

<comment> -> Comments starts with “#”.

<space> -> This is the space character.

Evaluation of Trip Programming Language:

Trip language will be evaluated in terms of 3 categories:

Readability: Our programming language can be read by everybody and understood with a little much effort. In terms of readability, there is not much effort needed because there are no shortenings that exist in the syntax and naming convention similar with C group languages.

Writibility: That language allows programmers to write in Trip Language easily. Syntax of our language is similar to C group languages which would help developers to adapt language easily.

Reliability: Since that language is both readable and writable easily, there is a lack of reliability in terms of security. However, since there is no ambiguity and once written programmes in Trip Language gives exactly the same output, we can mention reliability.

Hence, in terms of readability and writability we can easily say that those are the features of that language. However, when those two are present, reliability requirement is not fully satisfied.