Ex of code:

```
func findletter(string x, char c){
string nr = "";
int i = 0;
while (i < x.length){
        if x[i] == c;
        nr += i;
        break;
}
        else {
        i++;
}
rtrn nr;
}

mainPR{
string alphabet;
alphabet = "abc";
string result;
findletter (alphabet,c);
print (result)
}
```

Grammar design

Meta notations

Notations	Meaning
<x>	means that x is a nonterminal
x	means that x is a terminal i.e., a token or a part of a token
x^*	means zero or more occurrences of x
x^+	means one or more occurrences of x
|	separates alternatives

Table 1.Meta notations for the DSL Grammar

The DSL design includes several stages. First of all, definition of the programming language grammar $L(G) = (S, P, V_N , V_T)$:

- S – is a start symbol;
- P – is a finite set of production of rules;

- V_N – is a finite set of non-terminal symbols;
- V_T – is a finite set of terminal symbols.

V_T = { **FUNCTION**, **DO**, **SLEEP**, **REPEAT**, **USE**, **FOR**, **EACH**, **IN**, **END**, **A**, **B**, ... **Z** , **a**, **b**, ... **z**, **0**, **1**, ... **9**, **=**, **.**, **,**, **[,]** }

V_N = { <program>, <listOfCommands>, <basicCommands>, <initializationCommands>, <playCommand>, <sleepCommand>, <useCommand> , <repeatCommand>, <forEachCommand>, <functionCallCommand>, <naturalValue>, <floatValue>, <instrument>, <Time>, <variableName>, <listName>, <initializeFunction>, <initializeVariable>, <value>, <naturalList>, <floatList>, <functionName>, <functionBody>, <lowerCase>, <upperCase>, <digit> }

S = {<program>}

P = {

<program> → <listOfCommands>

<listOfCommands> → <basicCommands>$^+$
| <initializationCommands>$^+$
| <basicCommands> <listOfCommands>
| <initializationCommands> <listOfCommands>

<basicCommands> → <playCommand>
| <sleepCommand>
| <useCommand>
| <repeatCommand>
| <forEachCommand>
| <functionCallCommand>

<playCommand> → **PLAY** <naturalValue>
| **PLAY** <variableName>
| **PLAY** <functionName>
| <playCommand> <basicCommands>

<sleepCommand> → **SLEEP** <time>
| **SLEEP** <variableName>
| <sleepCommand><basicCommands>
| <sleepCommand><listOfCommands>

<time> → <floatValue>

<floatValue> → <naturalValue> **.** <naturalValue>

| <naturalValue>

<useCommand> → **USE** <instrument>

| <useCommand><basicCommands>

<repeatCommand> → **REPEAT** <naturalValue> **TIMES** <basicCommands> **END**

| <repeatCommand><basicCommands>

<forEachCommand> →

FOR EACH <variableName> **IN** <listName> **DO** <basicCommands> **END**

| <forEachCommand><basicCommands>

<functionCallCommand> → <functionName>

| <functionCallCommand> <basicCommands>

<initializationCommands> → <initializeFunction>

| <initializeVariable>

<initializeVariable> → <variableName> = <value>

<value> → <naturalValue>

| <floatValue>

| [<naturalList>]

| [<floatList>]

<naturalList> → <naturalValue> **,** <naturalList>

| <naturalValue>

<floatList> → <floatValue> **,** <floatList>

| <floatValue>

<initializeFunction> → **FUNCTION** <functionName> **DO** <functionBody> **END**

<functionBody> → <basicCommands>

< naturalValue > → <digit>$^{+}$

<variableName> → <lowerCase>$^{+}$ | <upperCase>$^{+}$ | _$^{+}$ | <digit>$^{+}$

<functionName> → <lowerCase>$^{+}$ | <upperCase>$^{+}$ | _$^{+}$ | <digit>$^{+}$

<listName> → <lowerCase>$^{+}$ | <upperCase>$^{+}$ | _$^{+}$ | <digit>$^{+}$

<lowerCase> → **a** | ... | **z**

<upperCase> → **A** | ... | **Z**

<digit> → **0** | ... | **9**

<instrument> → **piano** | **guitar**| **violin** | **drums**

}