COMS W4115
Programming Languages and Translators

Language Reference Manual –

skyEM

(Seungjin Kyunghwan Yonghan Exam Maker)

Seungjin Nam sn2119@columbia.edu
Kyunghwan Kim kk2367@columbia.edu
Yonghan Kim (Group Leader) yk2081@columbia.edu
1. Lexical Conventions

1.1 Comments
Two types of comments are used in skyEM.

**multi-line comments**: starts with characters ‘ /* ’ and terminated by ‘ */ ’.
**single line comments**: starts with characters ‘ // ’ and continues until the end of the line.

1.2 Identifiers
An identifier consists of digits, letters and underscore ‘ _ ’, and it is case sensitive.
The first character must be a letter or underscore.

1.3 Keywords
These words are reserved, therefore cannot be used as identifiers.

```
for    if    else    break    int
char   string    boolean    double    float
subject   load    ask    answer    choices
question   print
```

1.4 Numbers
Numbers consist of digits and for floating point numbers, optional decimal point ‘.’
can be used. Integer numbers and floating point numbers will be distinguished.

1.5 String Literals
String literals are anything enclosed by a set of double quotes ‘ ” ’. To have
double quotes inside the string, put backslash ‘ \ ’.

1.6 Characters
A characters can be a letter, number or a single symbol enclosed by a single quote ‘ ’.
1.7 Other tokens
There are a few reserved characters or pairs of characters that must be used
correctly in the language. The reserved characters and pairs are as follows.

```
{ } ( ) [ ]
+ ++ - -- * / %
== <= >= < > !=
```

2. Types
boolean : Boolean values that can either be true or false
int : standard 32-bit integers
float : 32-bit IEEE floating format
double : 64-bit IEEE floating format
string : a string of characters
char : a single character

3. Expression
3.1 Primary Expression
Primary expressions have identifiers, constants, strings, or expressions in
parentheses.

```
Primary-expression: identifier
| string
| (expression)
```

A parenthesized expression is a primary expression, which returns the value of
enclosed expression. The presence of parentheses makes the precedence higher.

3.2 Arithmetic Expressions
Arithmetic Expressions take primary expressions as operands.
Unary Arithmetic Operators
They are applicable to int, double, float. Unary operators ‘+’ and ‘-‘ can be pre-fixed to an expression. The ‘+‘ operator returns the expression in a positive form, whereas the ‘–‘ operator returns the expression in a negative form. The post-fix unary operators ‘++‘ and ‘--‘ are only applicable to int. ‘++‘ operators increment the operand, whereas ‘--‘ operators decrement it.

Multiplicative Operators
The binary operators ‘*‘, ‘/‘, ‘%‘ indicate multiplication, division, and modulo respectively. They are grouped left-to-right. They are only applicable to int, double, float.

Additive Operators
the binary operators ‘+‘, ‘-‘ indicate addition and subtraction, respectively. They are grouped from left to right. They are applicable to int, double, float.

3.3 Relational Expressions
Binary relational operators ‘>=‘, ‘<=‘, ‘==‘, ‘!=‘, ‘>‘ and ‘<‘ indicate whether the first operand is greater than or equal to, less than or equal to, equal to, not equal to, greater than, or less than the second operand, respectively.

3.4 Logical Expression
Logical operators take relational expressions as operands. Or operator ‘||‘ indicates the logical or of two relational expressions. It has the lowest precedence. And operator ‘&&‘ indicates the logical and of two relational expressions. It has higher precedence than or operator.

4. Statements
Statements are basic elements of the program. A sequence of statements are executed sequentially, unless flow-control statements indicate otherwise.

4.1 Statements in ‘{ ‘ and ‘ } ‘
If a group of zero or more statements can be surrounded by ‘{ ‘ and ‘ } ‘, then they are grouped and run together in a sequential manner.
4.2 Assignments
To assign right valued expression to left valued expression,

\[ \text{left-valued expression} = \text{right-valued expression}; \]

4.3 Conditional Statements
Conditional statements have two forms. The first form is,

\[
\text{if ( relational expression )} \\
\{ \\
\quad \text{statements;} \\
\} 
\]

The statement is executed if the relational expression is evaluated to be true. Otherwise, it skips the statements within the curly brackets and continues the program. The second form is,

\[
\text{if ( relational expression )} \\
\{ \\
\quad \text{statements;} \\
\} \\
\text{else} \\
\{ \\
\quad \text{statements;} \\
\}
\]

This form works same as the first form, but the relational expression is evaluated to be false, it executes the statements enclosed by else curly bracket.

4.4 Loops
All loops in the language take the form of a while loop. To use the while loop,
while ( relational expression )
{
    statements;
}

As long as the relational expression is evaluated to be true, the statements will be repeated.

Break statement is introduced to break out from the loop and continue the program. To use break statements, type keyword break followed by a statement terminator `;`

5. **General Function Calls**
Function call takes the following form,

    function_name(parameter list);

Function must be defined before it is called. Parameter list must take the same type of variables as defined previously.

5.1 **Return statements**
Return statement is used within the function definition body to return the specific value. A basic return statement would look like,

    return return_value;

5.2 **Include Other Files**
To attach other files,

    attach filename;

5.4 **Function Definition**
To define a function,
Function name is the name of the function that user has given and the parameter list is a list of identifier of arguments, separated by commas ‘, ’. Parameter list can be empty. Return type may be void.

### 5.5 Console output function

The function `print()` has two different versions. The first version takes a variable as an argument and prints out the value of the variable. It takes the form,

```c
print(variable);
```

The second version takes a string and prints it out to the screen. The string must be enclosed in double quotes and takes the form,

```c
print("This will be printed.");
```

### 5.6 File I/O Functions

Function `load()` will load a picture and display it or load a sound file and play it. Image file can be jpg and sound file can be mp3. To use `load()` function,

```c
load(file_type, filename);
```

### 6. skyEM Special Features

#### 6.1 Subject Object

This object represents one examination with all of its questions and answers along with other properties, such as time limit, order of the questions, and the types of questions. It is declared in the following way
subject subject_name(int, rand/seq(opt), int/all(opt))
{
    statements;
}

The first integer argument represents the time limit for this particular examination. The second argument will decide how the questions will be presented, either randomly or sequentially. This parameter is optional. The last argument determines how many questions will be appeared during the examination. If this is set to all, all questions will be appeared. This parameter is also optional.

6.2 Question Object
This object represents an individual question. It is created in the body of subject objects. There are several types of questions; multiple choice question, matching, true/false, one word answer and they are declared in the following manner;

    question multiple(int)
    {
        ask(string);
        answer(string);
        choices(string, string ...);
    }

Declaration of a question always takes an integer which is the points worth of this question. ask() function takes the questions string and displays it with an appropriate numbering scheme. answer() function takes the answer string and defines the answer. choices() function takes up to 6 answer choices strings and display them. Other questions objects are used in a similar manner.

6.3 Built in functions
There are several built-in scoring functions associated with subject object. First function is total() function, which returns total score of the examination. It is used as following,

    subject_name.total;
The second function is `score()` function, which returns the exam score for the subject. It is used as following,

```java
subject_name.score();
```

The third function is `percentage()` function, which returns the percentage value of the score. It is used as following,

```java
subject_name.percentage();
```

The fourth function is `start()` function, which starts the examination for the subject. It is used as following,

```java
subject_name.start();
```