LANGUAGE SPECIFICATION

***Group Members :***

***Winona Eleanor Kristen Fernandes (cs182032)***

***Iqra Muhammad Anwar (cs182015)***

***Hatim Khuzema (cs182003)***

**EPITOME LANGUAGE:**

Epitome is a trendy-motive system and user-friendly programming language with a C++ like syntax that compiles to local code and is case sensitive. As the name suggests, it was designed and inspired by the team members of this group and hence has unique keywords to differentiate it from other languages. Epitome is designed dependently as modules which are compiled one at a time. In Epitome language there will be some keywords, identifiers and operators that will help the user to solve algorithms in an easy way.

Epitome will prove as a hybrid that contains the functionality of the C++ programming language. This means that we will have all the features that are available in C++.

# IDENTIFIER:

An identifier is group of letters, digits and underscores. It starts with a letter and is case sensitive. The lexical aspect is free of syntax and semantic analysis. It takes each character as a token and disregards whitespace (space, tab, new line, returns and form feeds). Comments and remarks are also disregarded by the lexical analyzer. A comment is usually anything that’s written between //. Epitome supports single line comments.

A whole number consists of an arrangement of one or more integer value (i.e., 0123456789). There are no such thing as negative constants. To negate a negative number, we use the ‘- ‘unary operator.

A string constant is a series of zero or more printable characters or spaces enclosed between double quotes (“ “).

# VARIABLES:

# The rules of variables are defined as initially we have a datatype, then identifier then any operator or any string float value. This is the grammar/syntax designed for our function.

# OPERATORS:

The binary operators are + - \* / = <> < > <= >= & |while Parenthesis group expressions in the usual way. A leading – sign negates an integer constant. The binary operators >, <, >=, <= compare their operands. The operands need to be both integer or both string and produce the number 1 if the comparison holds or 0 if it does not. String comparison is done using ASCII lexicographic order.

The binary operators <>, = compare any two operands of the same type and return either an integer of 1 or 0. Integers are the same when they have the same worth, strings are equivalent when they have same characters that follow the same sequence.

The logical operators & and | are lazy logical operators. This is because they do not compare their right argument if comparing the left gives the result. Zero is false, everything else is true.

# KEYWORDS:

Iterative statements & conditional statements.

* **Iterative statements**
  + While loop
* **Conditional statements**
  + if
  + else
  + location (continue)
  + end (break)
  + return
* **Data Types**
  + iq (int)
  + wi (float)
  + ai (string)
  + wai (char)
* **Print & Read statements**
  + display (print)
  + read (take input)
* **Pre-processor directives:**
  + include
  + library (bits/stdc++.h)

**DATATYPES:**

**|** “int” | “float” | “string”

**DELIMITERS:**

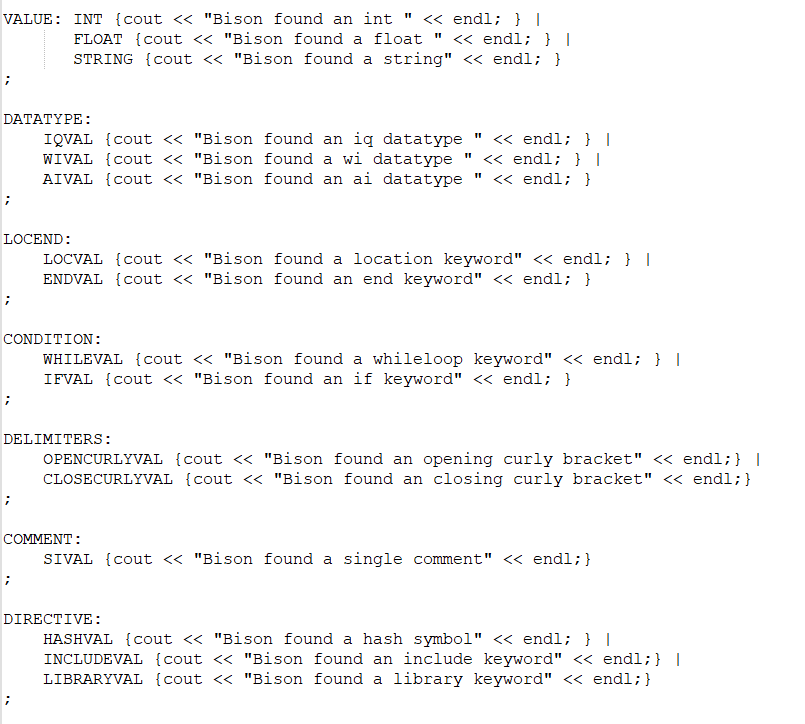
Delimiters are used to signify the end of a line (usually a ; ), an underscore (\_), curly braces {} and a comma (,). Delimiters are given a separate name to differentiate them from special characters.

**SPECIAL CHARACTERS:**

Few characters like @,#,$,^,&,% are recognized by the compiler as special characters. This means that only these characters are allowed to be used. However, some other characters like (#., +, - etc.) are used for other components like pre-processor directives or arithmetic operators.

**GRAMMAR:**

# 



# 