***LOOK#***

**LANGUAGE SPECIFICATION**

* **DATA TYPES:**

|  |  |  |  |
| --- | --- | --- | --- |
| DATATYPES | SYNTAX | EXAMPLES | CLASS PART |
| alpha | Alpha ID = constant. | Alpha a=<4>. | DT |
| digit | Digit ID = constant. | Digit a =2. | DT |
| word | Word ID = constant. | Word a = <<abc>>. | DT |

* **LOOPS**

|  |  |  |  |
| --- | --- | --- | --- |
| LOOPS | SYNTAX | EXAMPLES | CLASS PART |
| Till | Till (id relational operator id / const, id inc/dec) [statement] | Till(a< 5 , a++)  [statement] | Till |
| Untill | Untill(id relational operator id/constant>  [  Statement.  inc/dec.  ] | Untill(a==5)  [  statement  inc/dec  ] | Untill |

* **CONDITIONAL STATEMENTS:**

|  |  |  |  |
| --- | --- | --- | --- |
| DECISIONS | SYNTAX | EXAMPLES | CLASS PART |
| Case | Case(id relational operator id / const)  [statement]  Then Case [statement] | Case(a>5)  [statement]  Then Case  [statement] | Case |

* **INCREMENT/DECREMENT:**

|  |  |
| --- | --- |
| OPERATORS | CLASS PART |
| -- | INC\_DEC |
| ++ | INC\_DEC |

* **ASSINGMENT:**

|  |  |
| --- | --- |
| OPERATORS | CLASS PART |
| = | ASSIN\_OP |

* **LOGICAL:**

|  |  |
| --- | --- |
| OPERATORS | CLASS PART |
| && | LOG\_OP |
| ! | LOG\_OP |
| || | LOG\_OP |

* **RELATIONAL:**

|  |  |
| --- | --- |
| OPERATORS | CLASS PART |
| < | REL\_OP |
| > | REL\_OP |
| != | REL\_OP |
| >= | REL\_OP |
| <= | REL\_OP |
| == | REL\_OP |

**5. PANCTUATORS**:

|  |  |
| --- | --- |
| PANCTUATORS | CLASS PART |
| : | :  (terminator) |
| , | , |
| (  ) | (  )  (For conditional statements and loops) |
| [  ] | [  ]  (Body open ,body close) |
| ~ | ~  (In pointers) |
| ; | ;  (used in loops) |

**6. CONSTANTS:**

|  |  |  |  |
| --- | --- | --- | --- |
| CONSTANTS | R.E | DFA | CLASS PART |
| alphabets | '(Any Alphabet)' |  | CONST\_ALPHA |
| Digit | (digit)+ |  | CONST\_DIGIT |
| Word | "(Any Alphabet)\*" |  | CONST\_WORD |

**7. IDENTIFIERS/VARIABLES:**

Always starts with an alphabet

May have many alphabets

Can ends with a digit (only one)

Allowed: abc, abc9, s8,

Not Allowed: 9ed, 9e, s99

**R.E:**

(Letter)+(digit+null)

**STATEMENT TERMINATOR:**

|  |  |
| --- | --- |
| STATEMENT TERMINATOR SYMBOL | CLASS PART |
| : | : |

**COMMENT:**

|  |  |
| --- | --- |
| COMMENT SYMBOL | CLASS PART |
| # | # |
| \_#\_ ............. \_# | \_#  (multiline comment) |

**8. FUNCTION:**

|  |  |  |  |
| --- | --- | --- | --- |
| SYNTAX | EXAMPLE | CLASS PART/K.W | ANY ADDITIONAL NOTE |
| Rt func\_name(parameter)  [  Statement.  goback DT.  ] | DIGIT FUNC\_ADD(a,b)  [  Sum=a+b.  goback.  ]  (function definition typed at bottom of program above the END keyword) | FUNC\_1 | Function calling:  Sum =FUNC\_NAME (a, b).  Function declaration:  DIGIT FUNC\_NAME (a, b).  (written after start keyword) |

**9. ENTRY POINT OF CODE:**

|  |  |  |  |
| --- | --- | --- | --- |
| SYNTAX | EXAMPLE | CLASS PART/K.W | ANY ADDITIONAL NOTE |
| ProgramStart.  [  Statement.  ]  End. | ProgramStart.  [  Statement  ]  End. | ProgramStart  End. |  |

**MORE:**

In my language it is necessary to give space after typing anything (any word).

Pointers and classes added by instructor.

**Pointer:**

Pointer variable declared as ? followed by identifiers.

Eg. ?abc9

And assignment of address of a variable to pointer we use ~ operator.

eg:

~ab

A complete example:

Digit ab= 9 :

Digit ?abc9;

abc=~ab;

**Classes:**

Classes will b declared by the keyword Class

And an instance of a class will be instantiated by keyword Create.

Eg:

Class MyClass [ ]

MyClass MC = Create MyClass :