LANGUAGE SPECIFICATIONS

# BOOLE

بولی

SUBMITTED TO:

MA’AM AMNA IFTIKHAR

COURSE # 501

COMPILER CONSTRUCTION

BSSE-III (A)

. GROUP MEMBERS .

MUHAMMAD FASIH KHAN-B14158039

AHSAN JAWED -B14158003

## KEYWORDS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Aam | zati | Mehfuz | Naqsha | jaeza/khyal | sakin |
| Ewaz | brhana | Kro | jbtk | isme | ye |
| Lotao | koshish/dekho | pakro | Naya | brtr | badlo |
| Warna | warna-agr | agr |  |  |  |
|  |  |  |  |  |  |

## aam

‘aam’ is used as an access modifier. It resembles the ‘public’ keyword.

## zati

‘zati’ is used as an access modifier. It resembles the ‘private’ keyword.

## mehfuz

‘mehfuz’ is used as an access modifier. It resembles the ‘protected’ keyword.

## naqsha

‘naqsha’ resembles the keyword ‘class’.

## khyal

‘khyal’ resembles the keyword ‘abstract’.

## sakin

‘sakin’ resembles the keyword ‘static’.

## ewaz

‘ewaz’ resembles the keyword ‘default’.

## brhana

‘brhana’ resembles the keyword ‘extends’.

## kro

‘kro’ resembles the keyword ‘do’.

## jbtk

‘jbtk’ resembles the keyword ‘while’.

## isme

‘isme’ resembles the keyword ‘for’.

## ye

‘ye’ resembles the keyword ‘this’.

## lotao

‘lotao’ resembles the keyword ‘return’.

## koshish

‘koshish ’ resembles the keyword ‘try’.

## pakro

‘pakro’ resembles the keyword ‘catch’.

## naya

‘naya’ resembles the keyword ‘new’.

## brtr

‘brtr’ resembles the keyword ‘super’.

## badlo

‘badlo’ resembles the keyword ‘switch’.

## warna

‘warna’ resembles the keyword ‘else’.

## warna-agr

‘warna-agr’ resembles the keyword ‘else-if’.

## agr

‘agr’ resembles the keyword ‘if’.

# mamla

‘mamla’ resembles the keyword ‘case’.

## COMMENTS

* The single line comment is marked as //
* The multi-line comment is marked as <!—comment >.

## PUNCTUATORS

|  |  |  |
| --- | --- | --- |
| Symbols | Name | Purpose |
| ( ) | Parentheses | Used to contain lists of parameters in method definition and invocation.  Also used for defining precedence in expressions, containing expressions  in control statements, and surrounding cast types. |
| { } | Braces | Used to contain the values of automatically initialized arrays. Also used  to define a block of code, for classes, methods, and local scopes. |
| [ ] | Brackets | Used to declare array types. Also used when dereferencing array values. |
| ; | Semicolon | Terminates statements. |
| , | Comma | Separates consecutive identifiers in a variable declaration. Also used to  chain statements together inside a for statement. |
| -> | Arrow | Used to separate package names from sub-packages and classes. Also  used to separate a variable or method from a reference variable. |

## IDENTIFIERS

## Regular Expression:

L (L+d+\_)\*

## DFA:

L

L + D + \_

D + \_

## TRANSITION TABLE:

|  |  |  |  |
| --- | --- | --- | --- |
| S No. | L | D | \_ |
| 0 | 1 | 2 | 2 |
| 1 | 1 | 1 | 1 |
| dead | 2 | 2 | 2 |

## OPERATORS

## Arithmetic Operators

* “ + “ is used to add two values of same type and return the value of that type. For example: int = int1 + int2
* “ - “ is used to subtract two values of same type and return the value of that type. For example: int = int1 - int2
* “ \* “ is used to multiply two values of same type and return the value of that type. For example: int = int1 \* int2
* “ / “ is used to divide two values of same type and return the value of that type. For example: int = int1 / int2
* “ % “ is used to return the Reminder of division between two values of same type. For example: int = int1 % int2
* “ ^ “ is used to return same type Power of the values. For example: int = int1 ^ int2

## Bitwise Operators

* “ & “ is used to return the result of Logical AND between the bits of two entities. For example: Entity = entity1 & entity2
* “ | ” is used to return the result of Logical OR between the bits of two entities. For example: Entity = entity1 | entity2

## Relation Operators

* + “ == ” is used to relate two Equal Objects.
  + “ != “ is used to relate two Unequal objects.
  + “ <= ” , “ >= ” , “ > “ and “ < “ is used to determine the nature of unequal entities.

## Logical Operators

* + “ && “ is used to determine the bitwise AND result between two conditions.
  + “ || “ is used to determine the bitwise OR result between two conditions.

## CONSTANTS

## Integer:

## Regular Expression:

( + , - , null)d+

## DFA:

d

+ , - , \_

+ , - , \_

d

+ , - , \_

## TRANSITION TABLE:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S No | + | - | \_ | d |
| 0 | 1 | 1 | 1 | 2 |
| 1 | 3 | 3 | 3 | 2 |
| 2 | 3 | 3 | 3 | 2 |
| 3 | 3 | 3 | 3 | 3 |

## Character:

## Regular Expression:

L + /(n+r+t+a+/+b+f+’+”+0)

## DFA:

L

Escape char

\

Non-Escape char

## TRANSITION TABLE:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S no | ‘ | L | \ | Escape char |
| 0 | 1 | dead | dead | dead |
| 1 | dead | 2 | 3 | 2 |
| 2 | 4 | dead | dead | dead |
| 3 | dead | dead | 2 | 2 |
| 4 | Dead | Dead | Dead` | Dead |
| dead | Dead | dead | Dead | dead |

## String:

## Regular Expression:5

[ L + /(n+r+t+a+/+b+f+’+”+0) ] \*