**CS F301 – Principles of Programming Languages**

**Assignment 1**

Team Members:

Chitti Ankith Reddy – 2015B3A70534H

K.S.S.V Rama Krishna – 2015B3A70610H

Phani Shankar Ede – 2015B3A70420H

Aditya Addepalli – 2015B1A70719H

**Programming Language : D**

D is a general purpose systems programming language that has its roots built on the foundations of C and C++. Towards that end, the language can be seen as a collection of modules that can be compiled separately, or with libraries and compiled C code by a linker to create a native executable.

Apart from its array declarations and certain inbuilt data types, most of the syntax for D is very similar to C as it is often seen as an extension to C/C++.

<program> is used to indicate the start symbol for the language. We haven’t explicitly listed all of the inbuilt libraries available for D. They have been simply enlisted as <libraries>.(The standard library for I/O in D is the std.stdio).

In order to enhance readability, terminal symbols and certain other symbols such as the logical OR(‘||’) and round brackets, have been colored in red. ‘e’ indicates the empty string.

**Main Program:**

**<program>** -> <libraries> <declarations> void main() { <statements> }

**Variable Declarations:**

**<declarations>** -> <variable declarations><declarations> | <function declaration><declarations> | e

**<variable declarations>** -> <data type> <left dt><right dt> | <data type> <right dt>

**<variable>** -> <data type> <var name>

**<left dt>** -> <var name> = <expression>,<left dt> | <var name> , <left dt>|e

**<right dt>** -> <var name> = <expression>; | <var name>;

**<data type>** -> void | bool | char | dchar | wchar | byte | int | uint | long | ulong | short | ushort | float | double | real | ifloat | idouble | ireal

**<var name>** -> <name> | <array> | <dimensions> <array>

**<name>** -> <letter><string> | \*<name>

**<letter>** -> a|b|c…. |z|A|B|C…. |Z

**<string>** -> (<letter> | <digit> | \_)<string> | e

**<array>** -> <name><dimensions>

**<dimensions>** -> [<number>]<dimensions> | [ ]<dimensions> | e

**<number>** -> <digit><number>

**<digit>** -> 0|1|2|3|4|5|6|7|8|9

**Operations(With precedence order):-**

**<expression>** -> <expression> | <term1> | <term1>

**<term1>** -> <term1> ^ <term2> | <term2>

**<term2>** -> <term2> & <term3> | <term3>

**<term3>** -> <term3> + <term4> | <term3> - <term4> | <term4>

**<term4>** -> <term4> \* <term5> | <term4> / <term5> | <term5>

**<term5> ->** (<expression>) | <var name> | <function call> | <unary term> | <constant>

**<unary term>** -> ++<var name> | <var name>++ | --<var name> | <var name>--

**<constant> ->** <number> | -<number> | <number>.<number> | -<number>.<number>

**Function Declaration&Definition:**

**<function declaration>** -> <data type> <function type>

**<function type>** -> \*<function type> | <function name>(<parameters>); | <function name>(<parameters>){ <statements><return statement>}

**<return statement>** -> return (<var name>|<literal>); | e

**<parameters>** -> <left parameter><end parameter> | e

**<left parameter>** -> <data type> <name> | <data type> <letter><string> [<dimensions>],<left parameter> | e

**<end parameter>** -> <data type> <name> | <data type> <letter><string> [<dimensions>]

**Function Calling:**

**<function call>** -> <function name>(<call parameters>)

**<function name>** -> <letter><string>

**<call parameters>** -> <start call parameter><end call parameter> | e

**<start call parameter>** -> <var name>,<start call parameter> | <expression>,<start call parameter> | <function call>,<start call parameter> | e

**<right call parameter>** -> <var name> | <expression> | <function call>

**Sequential Statements:**

**<statements>** -> <variable declarations><statements> | <function call>;<statements> | <cond statement><statements> | <loop><statements> | <unary term>;<statements> | e

**Loop:**

**<loop>** -> <for loop> | <while loop> | <do while loop>

**<while loop>** -> while(<boolean expression>){<statements>}

**<do while loop>** -> do(<statements>)while(<boolean expression>)

**<for loop>** -> for((<variable> | <var name>) = <expression> | e; <boolean expression>|e ; <variable> = <expression> | <unary term>){<statements>}

**Conditional Statement(If Else):**

**<cond statement>** -> <if statement>|<switch statement>

**<if statement>** -> if(<boolean expression>){<statements>}<Else>

**<Else>** -> else <cond statement> | else{<statements>} | e

**<switch statement>** -> switch(<expression>){ <switch statements> }

**<switch statements>** -> case <literal> : <statements> <switch statements> | case <literal> : <statements>;break; | <switch statements> default : <statements>;break; | e

**<literal>** -> <name> | <number> | true | false | null | <number>.<number>

**<boolean expression>** -> <expression> | <expression><bool op><expression> | (<bool expression> || <bool expression>) | (<bool expression> && <bool expression>) | !(<bool expression>)

**<bool op>** -> == | != | >= | > | <= | <