

**SCHOOL OF ADVANCED TECHNOLOGY**

**ICT - Applications & Programming**

**Computer Engineering Technology – Computing Science**



# A40

Language Specification (Full Compilation)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Team:** | | | | | | | | |
|  | **[Jwell Lalakiya]** | - Id: | **[041056229]** | **/** | **[Priya Patel]** | - Id: | [041084491] |  |

|  |  |  |
| --- | --- | --- |
| **Language Name [** | **DefaultX** | **]** |

***This assignment is optional but once correctly developed, bonus marks (5pt) will be given. This template is suggested (not mandatory) to answer A40 Specification1.***

1 This assignment requires that the entire documentation previously developed (A11, A21, A31) must be reviewed and updated to be conform with the latest version of the code / project developed. The simple copy about what has been submitted in the previous assignments, will not give any marks.

|  |  |
| --- | --- |
| ***Part 1*** | ***Informal Language Specification*** |

 **[Basic Instructions]** Review your **A11** to make compatible with the latest definitions.

|  |  |
| --- | --- |
| ***1.1.*** | ***User Manual*** |

**Element 1: Name / Extension**

**Name:** DefaultX

**Extension: “**.dfx”

**DefaultX** is an open-source programming language that is similar to programming language “Go”.

**Element 2 – Comments**

**Single Line Comment:** (/ /)

**Multi Line Comment:** (//\*.….\*//)

**Element 3 – Keywords**

"int32",

"float32",

"string",

"break",

"default",

"func",

"case",

"if",

"else",

"switch",

"return",

"for",

"while",

"void"

**Element 4 – Datatypes**

**Integer:**

* Syntax: int
* Size: 4 bytes (32 bits)

**Strings:**

* Syntax: string

**Boolean:**

* Syntax: bool
* Size: 1 byte

**Element 5 – Variables**

**Integer:** var [variable name] **int** = [integer value].

**Boolean:** var [variable name] **bool** = [true/false]

**String:** var [variable name] **string** = [string value]

**Floating:** var [variable name] **float32** = [single precision value]

## Element 6 - Commands

**Attribution / assignment**:

* A programmer can assign a value to any variale using “**Assignment opperator (=)**”.
* Our language does support casting but it will be only limited to similar type. For example, a programmer cannot cast from “**int**” to “**string**”.
* All the arithmetic operation will be performed using regular **arithmetic operators** i.e. **(+,-,/,%,\*)**
* You can join two strings using “**+**” operator if both the operands are string.

**Selection:**

In our language conditions there are two conditional statements:

* **If/else:**

**if (condition)** {

//block of code

} **else if (condition)** {

// block of code

} **else** {

//block of code

}

* **Example:**

**if age < 18 {**

**println("You are a minor.")**

**} else if age >= 18 && age < 65 {**

**println("You are an adult.")**

**} else {**

**println("You are a senior citizen.")**

**}**

**Conditions** usually work when there are two operands and one comparison operator in between. So basically, the operands are compared bassed on the comparison operator used in between. Following are the comparision operator that can be used in our language:

* “==” / “!=”
* “>”/”<”
* “>=”/”<=”

Programmer can also combime two conditions using logical operators. Following are the list of logical operators:

* AND
* OR
* NOT

**Interaction:**

Our language has **for loop** as well as **for each** loop to loop through array elements.

**for** ([*initialize loop variable*];[*exit condition*];[*increment/decrement*]){

}

**Input**:  
Programmers can use “**scanf**” to read an input from the keyboard depending on the format specifier.

***s*canf(“[*format specifier*]”,&[*vairable name to be stored in*]);**

**Output:**

Programmers can use “**println**” to read an input from the keyboard depending on the format specifier.

***p*rintln(“[*String*/*format specifier*]”,[*vairable name the value is stored in*]);**

|  |  |
| --- | --- |
| ***1.2.*** | ***Real Examples*** |

*[Give some examples that can work correctly in your front-end compiler (specially, the parser)]*

**# Example1**

**func main() {**

**var sum int**

**for i := 0; i < 5; i++ {**

**sum += i**

**}**

**println("Sum:", sum)**

**}**

**# Example2**

**int main() {**

**printf("Hello, World!\n");**

**return 0;**

**}**

|  |  |
| --- | --- |
| ***Part 2*** | ***Formal Language Model (RE, Automaton)*** |

***2.1. Define the RE elements***

## Element 1: Lexeme Classes

**L (Letters): A-Z | a-z**

**D (Digits): 0-9**

**S (Double Slash): //**

**J (Multi Line - Start): //\***

**K (Multi Line – End): \*//**

**E (End line): \n**

**U (Under Score): \_**

**P(Period): .**

**Q(Double Quote): “**

**M(Paranthesis): (**

**O(Other): ^[LDSJKEUPQM]**

## Element 2: Define the RE for main kinds of tokens

**\* Comments: (Single Line) SC=** S(^S)\*E**, (Multi Line) MC=** J(^J,K)\*K

**\* Variables: VAR=** L(L|D|U)\*

**\* Methods: MID=**L(L|D|U)\*M

**\* Literals:**

**- Integers: IL=** D+

**- Float: FL=** D+PD+ | PD+

**- String: SL=** Q(^Q)\* Q = Q(LDSJKEUPQM)Q

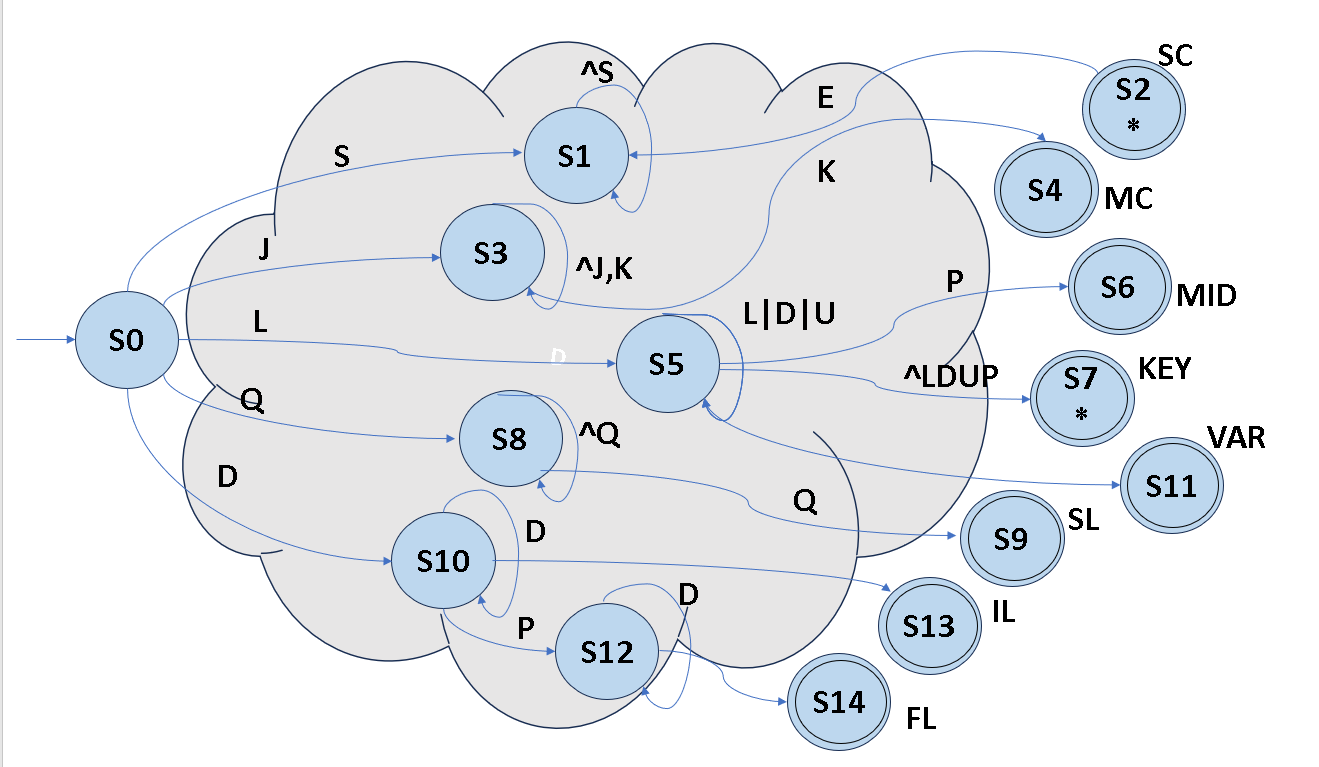
## Element 3: Keyword lists

**break, default, func, case, struct, if / else, package, switch, const, type, continue, for, import, return, var, main&, print&, input&**

***2.2. Automaton***

## Draw / copy image of the automaton (use the latest version of your Scanner)

**# Automaton #**



|  |  |
| --- | --- |
| ***Part 3*** | ***Formal Language Specification / Grammar*** |

|  |  |
| --- | --- |
| ***3.1.*** | ***Define the Language BNF*** |

## Program (“Start” production)

*[Define the program]*

**<program>**  **<func\_name>()** {

<data\_session>

<code\_session>

}

**<func\_name>**  **ID\_T LPR\_T**

*[Continue with all Grammar Derivations – Solving problems such as “Left-Recursion” and/or “Left-*

*Factoring”]*

## Marking Rubric[[1]](#footnote-1) (5 pt)

|  |  |
| --- | --- |
| **Maximum**  **Deduction (%)** | **Deduction Event** |
| - | **Plagiarism:** |
| Check | 3-strike policy3 (**AA32**, **SA07** and **IT01**) |
| Incompatible definitions | Definitions in parts I, II or III are not consistent (ex: different keywords/notations) |
| - | **Assignment Elements:** |
| Up to 1 pts | PART I: Basic Language |
| Up to 2 pts | PART II: Models (RE / Automaton) |
| Up to 2 pts | PART III: BNF Language |
| **Final Mark** | **Formula: 5**\*((100- ∑ penalties + bonus)/100). |

**File update**: Nov 25th 2023.

**Good luck with A40!**

1. If the documentation submitted has not been updated to the latest version of the code/project, **no marks** will be given. 3 The plagiarism detection will imply in the “3-strike” policy: starting with ZERO, then moving to course failure or program cancelation (see the Algonquin College documents: [https://www.algonquincollege.com/policies/)](https://www.algonquincollege.com/policies/). [↑](#footnote-ref-1)