



---

## SCHOOL OF ADVANCED TECHNOLOGY

ICT - Applications & Programming  
Computer Engineering Technology – Computing Science

---



# A31

## BNF - Grammar

Language Specification

*A31 Specification*



## ASSIGNMENT 3.1 – LANGUAGE GRAMMAR (used in **Parser**)

### General View

**Due Date:** prior or on **Jul 21<sup>st</sup> 2024 (midnight)**

- **2<sup>nd</sup> Due date** (until **Jul 28<sup>th</sup>**) - **50%** off.

**Earnings:** **5%** of your course grade.

**Development:** Activity can be done **individually** or in teams (**only 2 students** allowed).

**Purpose:** Define the BNF of **your language**.

- ❖ This is an important activity from front-end compiler that is based on the definition of your language.
- ❖ Your next activity (*parser implementation*) will be based on the definitions given for your language.
  - Start reviewing and **fixing** your definitions done in **A11** (Language Proposal).
  - Then, continue **defining all the grammar** for your language: define the non-terminals and terminals of your language.
- ❖ **MAIN IDEA:** Use the section **Sofia Model Definition** (**A31\_S23\_YOURLANGUAGE\_Template**) to answer your assignment correctly.
  - Change the document to describe **your language**.
  - You can remove elements not used in **your language**.
  - You can include new elements if necessary (for instance, if you are including new datatypes).
  - What does matter is the final consistency between your BNF and the language that you are creating.

- **TIP:** One strategy is checking the code examples that you are using for each assignment and check if they are obeying the rules that you are defining in your grammar.

#### NOTE 1:

Your language **can be updated** to prepare it better for the implementation. But be careful about some effects.

- *For instance, if you are changing the tokens (replacing), your scanner – see the tokenizer function and your transition table – should reflect. Otherwise, you will have a difference version of the grammar that you can recognize between scanner and parser.*

#### Task 1: BNF (5 marks)

See the [A31\\_S23\\_YOURLANGUAGE\\_Template](#) document that defines the **Sofia** language (or, eventually, check the **Appendixes** in the **Lecture Notes**). You need to create your own language grammar.

- Start defining instructions to define the syntax based on some **basic elements**: keywords, comments, etc.
  - Use the tokens that you have defined in the **scanner** (for instance, INL\_T for integer literals, etc.)
- Define correctly **syntax**, including elements of your program, the statements, etc..

**TIP:** Your language can be reviewed / updated. What does matter is that you can define your own specification, that must be different from **Sofia** language.

- Be sure that all elements of your grammar are there.
- Also check the compatibility with your specification and what you have previously defined.
- **IMPORTANT:** In the template, you just need to include the grammar for your language, you **do not need** to solve these two problems:
  - **LR - Left recursion** (avoid the recursion without prefix)
  - **LF - Left factoring** (your language must have one prefix).
- You also don't need to define the **FIRST** set (the first terminals that will be used in the grammar during the implementation).

#### NOTE 2:

Sometimes, you can start defining rules using LR or LF. However, you need to solve when you go to the implementation ([A32](#)).

- *If you do not solve these problems, your parser will not be able to work in the end.*

## How to Test

The basic tests that you need to test in your BNF are related to the following cases:

- *One method for “Hello world”*
  - *One code to use variables (see for instance, the mathematical expression in the following example);*
  - *Utilization of Inputs and outputs (including string messages).*
- Change this file (starting with the **name** of your language) and check all BNF rules described here, adapting it to your language.
  - **Example:** How to calculate the volume of a sphere (the mathematical formula is:  $V = \frac{4}{3} \pi r^3$ ).
  - Showing a **Sofia** example:

```
# Sofia Example (Volume of a sphere) #
main{
    data {
        real PI%, r%, Vol%;
    }
    code {
        PI% = 3.14;
        input&(r%);
        Vol% = 4.0 / 3.0 * PI% * (r% * r% * r%);
        print&(Vol%);
    }
}
```

Ex: R Language

```
# Sphere Volume
sphereVolume <- function(radius) {
  (4/3) * pi * radius^3
}
radius <- 5.0 # Replace this with the desired radius
volume <- sphereVolume(radius)
cat("Volume of the sphere with radius", radius, " is ", volume, "\n")
```

## Submission Details

- ❖ **Digital Submission:** **Compress** into a **zip** file with **ALL files** that you are using in this model – essentially, DOC file, but you can eventually include pictures. Also include a cover page.
- ❖ The submission must follow the course submission standards. You will find the Assignment Submission Standard as well as the Assignment Marking Guide ([CST8152\\_Compilers\\_ASSAMG.pdf](#)) for the Compilers course on the Brightspace.
- ❖ Upload the **zip** file on Brightspace. The file must be submitted prior or on the due date as indicated in the assignment.
- ❖ **IMPORTANT NOTE:** The name of the file must be **Your Last Name** followed by the last three digits of your student number followed by your lab **section number**. For example: [Sousa123\\_s10.zip](#).
  - If you are working in teams, please, include also your partner. For instance, something like: [Sousa123\\_Melo456\\_s10.zip](#).
  - **Remember:** Since we have just one lab professor, students from the **different sections** can constitute a team.
- ❖ **How to Proceed:** You need to demonstrate your progress to your Professor in **private Zoom Sections** during Lab sessions.
  - If you are working in teams, **you and your partner** must do it together, otherwise, only the student that has presented can get the bonus marks.
  - **Eventual questions** can be posed by the Lab professor for any explanation about the code developed.
  - Each demo is related to a **specific lab** in **one specific week**. If it is not presented, no marks will be given later (even if the activity has been done).

## Marking Rubric

Maximum Deduction (%)	Deduction Event
-	<b>Plagiarism:</b>
Check	3-strike policy <sup>1</sup> ( <b>AA32</b> , <b>SA07</b> and <b>IT01</b> )
-	<b>Severe Errors:</b>
2.5 pt	Late submission (after 1 week due date)
5.0 pt	Missing demo (zero <sup>2</sup> )
-	<b>Assignment Elements:</b>
2.5 pt	Missing demo (50% deduction)
<b>Task 1</b>	<b>Language BNF</b>
Up to 5 pts	Syntax Definition
Up to 2 pts	Correctness / Completeness
Up to 2 pts	No left-recursion, no left-factoring
Up to 2 pts	Compliance with examples provided
<b>ADDITIONAL</b>	<b>Small problems</b>
Up to 1 pt	Language adaptation (missing elements – ex: datatypes / constants)
Up to 1 pt	Unjustified modification (if you changed the language, explain why)
Up to 1 pt	Other minor errors
1 pt	Bonus: GitHub utilization
-	<b>Bonuses</b>
Up to +1 pt	Bonus: original ideas developed by language.
<b>Final Mark</b>	Formula: $5 * ((100 - \sum \text{penalties} + \text{bonus}) / 100)$ , max score 7%.

### Final Message

Remember that your language (your city name) must have a proper grammar (different from **Sofia Language**). Remember to provide the inputs that you are using – they will be especially necessary to the next (and final) assignment.

**File update: Jul 1<sup>st</sup> 2024.**

**Good luck with A31!**

<sup>1</sup> 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/>).

<sup>2</sup> If a course requires demos, they are not optional. If a student **does not demo** their work, they should receive a **grade of 0** on that assessment, not a grade reduction.