**Name of the programming language**: *Najim*

**Team Members**: Abdulrahman Zaiter, Brendan Jones

**Introduction**:

*Najim* is a programming language that is built over *JAVA*. It brings more simplicity into writing code and helps students who are new to coding to learn and apply without worrying about the complexity of Object Oriented languages. The inspiration of calling the language *Najim* comes from the beauty of the corresponding word from Arabic to English as “Star”. Najim brings a state of the art support for Indian numerals in addition Arabic numerals in the code, which makes it the first known language written over *JAVA* that does such features. *Najim* also support complex string arithmetic operations such as string duplications and concatenation through simple expressions. It also removes the confusion of *JAVA* in octal numerals as they will be notated as “0o<octal\_num>” instead of “0<octal\_num>” which brings a lot of confusion to the *JAVA* programming language. Moreover, it supports arithmetic powers through simple operator “\*\*” instead of calling a specific utility function to do so.

*Najim* source code “.nj” will be a file written with any text editor. Implicitly, *NajimC* (compiler), will translate the source code into *JAVA* code or *JVM* code (under research) and produce a runnable application that the user can test and use. The main class in the *JAVA* code will be defined as the file name for ease of use as the coder won’t have to specify that explicitly. The main function is called start() and this is where the execution pointer starts executing statements from, coder can specify whether they want to implement exit() function as it will be called once the program has ended.

**EBNF Notations:**

In *Najim’s* EBNF notations, we followed the proper valid syntax that is not provided in the class slides. The basic idea of them is to define a protocol for *Najim’s* programming language that the coders should follow to produce a proper working code. In addition to that, and since we’ve used a valid EBNF syntax, we might use it in our lexical analysis and parsing stages. Through *Najim’s* BNF, we can simply refer any entity whether it is virtual or real to the syntax and semantics of the language in a readable manner.

**Kindly look at the file “Language.ebnf” for all Najim’s EBNF notations.**