Universidad de los Andes

ISIS-4217 Paradigmas de programación
Functional programming
Date: October 6, 2025

Hand-in each of the following tasks on an independent `.oz` file. Make sure of following the functions specification as well as the file name conventions. Take into account, your code will be machine graded, if it does not follow the specification, it will be marked as a non-compiling program (a 0). Additionally, make sure to test your code before handing it in, if your code does not compile or run, it will be graded as a zero.

Task 1. Consider the program in Snippet 1, to represent an expression language. By definition expressions are made out of numbers, to which we can apply operations, for example `Sum` (the sum of two expressions). The program should be able to evaluate and print expressions.

Extend the program to add new operations to expressions (`Difference`, `Multiplication` , `Modulo`), following the same structure as `Sum`. Additionally, add new functionality for expressions, `ToString`, that describes an expression but as a string (For example, the expression `3+4` would become `"three plus four"` after turning it into a string. In the case of `Difference` you should print the string `"minus"`, in the case of `Multiplication` you should print the string `"times"`, and in the case of `Modulo` you should print the string `"modulo"`) (we will only use numbers up to 999)

```
%atoms defining expression types. num sum ...
%ExpPrint : Expression -> String
proc {ExpPrint num I}
    {Show I}
end
proc {Print sum Left Right}
    {System.showInfo Left#"+"#Right}
end

%Eval : Expression -> Int
fun {Eval num I}
    I
end
fun {Eval sum Left Right}
    Left + Right
end
```

Snippet 1: Task 1.

Hand-in a file `language.oz` with your solution.

Universidad de los Andes

ISIS-4217 Paradigmas de programación
Functional programming
Date: October 6, 2025

Task 2. Write an functional program that represents a square matrix an may operate sums and products over the matrix. Complete the implementation given in the file `matrix.oz`. **Hand-in the file matrix.oz with all the requiere functions correctly implemented, you can add any auxiliar functions you may need.**

Task 3. Mastermind The mastermind game consists of a subset of four colors out of six possible colors to choose from, defined as the secret code by the codemaker. Given a code, the codebreaker can query the codemaker with an array of four guesses for the colors. The codemaker then responds with at most four colors answering the correct guesses. A black answer is given for each color guessed in the correct position. A white color is answered for each correct color not in the correct position.

The code must be found by the codebreaker within 12 tries.

Hand-in the file mastermind.oz with all the requiere functions implemented, you may add any auxiliar function you need.