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 not be graded. Additionally, make sure to test your code before handing it in, if your code does not compile or run, it will not be graded.

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 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)

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

Task 2. Write an OO program that represents a square matrix (a `Matrix` object). Complete the implementation given in the file `matrix.oz`. **Hand-in the file matrix.oz will all the requiere method implemented, you could add any auxiliar method or function.**

Task 3. Mastermind The 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 method implemented, you could add any auxiliar method or function.

```
class Expression
    meth print
        {System.showInfo "Base method does nothing"}
    end
    meth eval(R)
        {System.showInfo "Base method does nothing"}
    end
end

class Num from Expression
    attr n:0
    meth init(Val)
        n := Val
    end
    meth print
        {System.showInfo @n}
    end
    meth eval(R)
        R = @n
    end
end

class Sum from Expression
    attr left right
    meth init(L R)
        left := L
        right := R
    end
    meth print
        {@left print} {System.showInfo "+"} {@right print}
        %for module print "mod" as the operator
    end
    meth eval(R)
        local LR RR in
            {@left eval(LR)}
            {@right eval(RR)}
            R =  LR + RR
        end
    end
end
```

Snippet 1: Task 1.