Team: Gocaml – Olivier Melancon and Terrence Ko

# Milestone 2 report

**This milestone was submitted two days late without penalty with permission from Laurie**

Data structure for typechecking

- We initially tried using an immutable data structure for holding the symbols but it proved difficult so we implemented a stack of lists.
- Each list contains the symbols declared within a scope.
- We don't use hash tables because they take much longer to implement than lists and given that our programs aren't particularly large, list traversal isn't very detrimental.
- Ocaml's pattern matching is also much better suited for playing with lists instead of hash tables.
- When encountering a function, a new list is pushed onto the stack for the scope.
- A non-destructive pop is used for looking up the type of a symbol.

Typechecker help from the weeder

- The typechecker returns an annotated AST which is useful for printing the types alongside the code in the pretty printer.
- Basic typechecking is done in the weeder, that is an error is raised when an expression is assigned something but the said expression is cannot be assigned to.
- On assignation in list, the weeder also checks that assignees and expressions are one-to-one

Icebox

- Binary operators need to have the same type on both sides (for now)
- Type casting is checked, can cast floats to int and vice versa, rune can be cast to string but not the other way around.

Work share

- Olivier has worked on prettyprint, the weeder, completing the ast and sticking everything together.
- Terrence has done the totality of what is currently available in the typecheker.