Summary of Differences between Goo and Go

A Goo program consists of only a single file (whose name has the '.go' suffix). That program has access to only a few necessary library functions.

What's Included in Goo?

These simple numeric datatypes: `int32  uint32  int64  unit64  float32  float64`

strings (with UTF8 character coding) – the interpreted kind only, runes (implemented as `int32`), and booleans.

These composite datatypes: arrays and slices (one dimensional only), structs (with no anonymous fields or tags), pointers.

Executable statements with these forms: blocks, if statement, for loop, assignments, function calls, return statements, inc/dec statements, break and continue statements, goto statements.

Declarations of these kinds: const declarations, type declarations for struct types only, variable declarations, short variable declarations function declarations (no variadic signatures can be declared and the function body must be provided).

Expressions and values of these kinds: qualified identifiers to access library package functions and fields of structs, composite literals for initializing arrays and structs, array/slice indexing, expression operators containing all the unary and binary operators (except for '`<-`'), address operator and dereference operator, string concatenation.

Conversions from one datatype to another.

The `len` and `cap` functions for use with arrays and slices.

What's NOT Included in Goo?

Imaginary numbers!

Interfaces, map types, channel types.

Statements of these kinds: switch, select, send, type switch, go, defer, fallthrough.

Methods and method declarations.

Function literals.

The receive operator ('`<-`')

Note

Although the Goo compiler's front-end will, after Assignment 1, be able to parse the constructs listed above, there may be too many constructs to implement in the time available. The later stages of the compiler may have to generate messages of the form "sorry, the foobar construct is not supported".