GUIDELINES FOR COMPILERS PROJECT

Each team will have a distinct vision of what their language should be. This is ok. So I will not have a rigid set of constraints on the final outcome. Ultimately, the language you build should be something that is usable to solve any competitive programming question.

There are some broad design categories that can be applied to any language. If it does not, contact me. Here are some categorizations:

- Low-level (C-like) vs High-level (Python-like)
- Statically- vs dynamically typed
- Imperative/OOP vs Functional

Below, I list possible features of languages. Some of these features are MUST, that is, any language must have them. Then, there are DESIRABLE features. These are optional.

MUST

- Support primitive data such as numbers, strings, and arrays or lists.
- Support conditionals such as if-else or match or cond.
- Support loops via for, while, or recursion.
- Support functions.
- Support compilation into bytecode/machine-code.
- Automated test coverage of 100%.

DESIRABLE

- Support for lexically-scoped variables.
- Support for a dictionary type.
- Support user-defined types.
- Support recursive pattern matching.
- Good error messages that point out location and suggest fixes.

MUST FOR LOW-LEVEL

- Support for controlling layout of data down to the bits.
- Function inlining and peephole optimizations.

MUST FOR HIGH-LEVEL

- Support for first-class functions.

MUST FOR STATICALLY TYPED

- Type errors before evaluation.

MUST FOR DYNAMICALLY TYPED

- REPL that does not crash on errors.

MUST FOR IMPERATIVE

- Support for loops.
- Support for mutable data.

MUST FOR FUNCTIONAL

- Support for proper closures (See "Man-or-boy" test).
- Support for tail-call elimination optimization.