The Extended-Kaleidoscope Project - Compiler Requirements

Version 0.2 - 2018-01-12
Hal Finkel and Kavon Farvardin

Your project must build and run from a fresh clone of your repository. If any additional steps, such as initializing git submodules, is required, you must document this in your README file.

** C/C++ Submissions **

Your project must use one of the following build systems:

1. Make
2. CMake

Note that CMake is a cross-platform build system, which can generate build configurations for a number of IDEs and build systems (see here for a complete list: https://cmake.org/cmake/help/latest/manual/cmake-generators.7.html).

CMake tutorials:
http://derekmolloy.ie/hello-world-introductions-to-cmake/
https://cmake.org/cmake-tutorial/

The default target must build your compiler. There's also nice docs on other useful targets here: https://www.gnu.org/prep/standards/html_node/Standard-Targets.html]]

** Python Submissions **

Your Python submission must run with the version of Python on linux.cs.uchicago.edu, currently v2.7.12 (or v3.5.2).

If your project has any dependencies on libraries installed on your machine via the pip package manager, you must include a Requirements File at the root of your project. You can generate a Requirements File using:

```
pip freeze > requirements.txt
```

(see here for more information: https://pip.pypa.io/en/stable/user_guide/#requirements-files)

** Interface **

The compiler should be named ./bin/ekcc or ./bin/ekcc.py and should permit usage as:

./bin/ekcc[.py] [-h|-?] [-v] [-O] [-emit-ast|-emit-llvm] -o <output-file> <input-file>

Where -h or -? should produce some help/usage message and the names of the authors.
Where -v puts the compiler is "verbose" mode where additional information may be produced to standard output (otherwise, for correct inputs, no additional output may be produced, except for lines beginning with the string "warning: ").
Where -O enables optimizations.
Where -emit-ast causes the output file to contain the serialized format for the AST
Where -emit-llvm will cause the LLVM IR to be produced (unoptimized, unless -O is provided).
Where -o <output-file> names the output file.
Where <input-file> names in input source code.

For correct inputs, the exit code of the compiler must be 0. For incorrect inputs, the error code must be non-zero. For incorrect inputs, the compiler should produce at least one line of explanation for the user. All such lines should start with the string "error: ".

** AST Format **

The -emit-ast option shall emit the abstract syntax tree for the input program YAML format. For information on YAML format, see:
http://docs.ansible.com/ansible/latest/YAMLSyntax.html
http://www.yaml.org/

The output should look something like the following example. Please make sure that your output will parse as valid YAML (http://www.yamllint.com/ may help).

```
---
prog:
  -
    what: extern
    type: int
    globid: getarg
    tdecls:
      - type: int
  - what: func
    type: void
    globid: foo
    tdecls:
      - type: int
      - type: ref int
  - what: func
```

```
    type: int
    globid: bar
    blk:
      - what: decl
        name: $x
        init: 1
      - what: decl
        name: $y
        init: 5
      - what: return
        exp:
          what: binop
          op: +
          lhs: $x
          rhs: $y
...
```