*Safi Nassar - assignment p3*

# Usage:

- `make clean && make testclean && make`
- `make test`

Executing the test will generate output file of the formated program.

**Prerequisites & dependencies:**

- OpenJDK 11
- Java Cup
- JLex

# Tests documentation:

Input file test.cminusminus should be compared with formatted result in test.out

---

# Java Cup Parser for C– language:

- The parser checks input programs for syntax errors, by building an AST representation of the input stream, and then unparses to display a textual represntation of the syntax directed translation.
- [x] https://www.cs.princeton.edu/~appel/modern/java/CUP/manual.html
- [x] Download Java CUP
- [ ] Setting up JLex/JavaCUP and $CLASSPATH (for Linux and Mac) http://youtu.be/6bZIsOJFbrE
- [ ] Setting up Java SDK, JLex/JavaCUP, and $CLASSPATH (for Windows) http://youtu.be/ZS7_FPmd1II
- [ ] Examples - http://www2.cs.tum.edu/projekte/cup/examples.php

## Tasks

- Files to edit:
  - [x] parser specification (cminusminus.cup using .grammar).
  - [x] unparse methods for the AST nodes (in ast.java).
  - [x] input file (test.cminusminus) to test implementation.

- Steps:
  - [x] Add a few grammar productions to cminusminus.cup - start by making a very small change to cminusminus.cup.

  - [x] Write the corresponding unparse operations - update the appropriate unparse method in ast.java, and check if parser works.
  - [x] add the rules needed to allow struct declarations.
  - [x] add the rules needed to allow programs to include functions with no formal parameters and with empty statement lists only, and update the corresponding unparse methods.
  - [x] Write a test program that uses the new language constructs.
  - [x] Create a parser (using make) and run it on your test program.
  - [x] add the rules (and unparse methods) for the simplest kind of expressions – just plain identifiers.
  - [x] statement nonterminals
  - [x] cminusminus.cup adding rules for the statements, and ast.java to define the unparse methods needed for those statements.
  - [x] Write test inputs statements
  - [x] makes sure parser and unparser work on statements
  - [x] expression nonterminals implementation. need to give the operators the right precedences and associativities.
  - [x] remaining productions that need to be added
  - [x] add test cases for final version of test.cminusminus
  - [x] do not try to implement all of nonterminals at once. Instead, add one new rule at a time to the Java CUP specification, make the corresponding changes to the unparse methods in ast.java, and test your work by augmenting your test.cminusminus or by writing a C– program that includes the new construct you added, and make sure that it is parsed and unparsed correctly.
  - [x] use unparse output as input to parser to check for errors.

## Submission:

- [x] Create pdf from markdown: `pandoc README.md -o <lastname.firstname.Pn.pdf>`
  - [ ] generate markdown from javadoc and remove redundant comments
    or
  - [ ] generate javadoc to extract method headers: `find . -type f -name "*.java" | xargs javadoc -d ../javadoc`
- [ ] Add headers for each file
- [ ] Verify code format
- [ ] Verify code execution on CSL machines
- [x] Handing in: ensure that you do not include any extra sub-directories. lastname.firstname.lastname.firstname.P3.zip +—+ deps/ +—+ ast.java +—+ cminusminus.cup +—+ cminusminus.grammar +—+ cminusminus.jlex +—+ ErrMsg.java +—+ Makefile +—+ P3.java +—+ test.cminusminus +—+ lastname.firstname.lastname.firstname.P3.pdf