# COP-4020 PROGRAMMING LANGUAGES CONCEPTS

## ****Project: SUBC Parser**** DUE: End of the semester

**Remember:**

\* Have your name and UF ID on what you submit

\* Submit everything in a single tar file to Canvas.

\* Develop the software on any platform you wish, but you \*MUST\* port it to C++ on

thunder.cise.ufl.edu (gcc 4.8.4 on Ubuntu 14.04.5 LTS, as of 10/20/2016). We will

compile it, run it, and grade it there, and nowhere else.

Goals:

The objective of this first project is to implement a lexical analyzer, and a recursive descent parser for a (sort of ) subset of C language called SUBC.

The grammar for the language is in file Project\_Grammar.docx.

You must write your OWN lexical and syntax analyzers in C++. You are NOT allowed to use 'lex' and 'yacc' or any similar software in order to implement this project.

The lexical rules for SUBC are in file Project\_Lex.doc.

Regarding grading and submission:

1. **Submit your parser’s source code, in a single tar file, via Canvas.**

**The tar file must unpack the files into the current directory, not some directory**

**below the current one, allowing us to perform the sequence shown in Part 4 below.**

**Put in a BUGS file all comments of known errors (if any) of your project.**

It is MUCH better for you to list the limitations of your software than to let us discover

them. The more time it takes for us to find what's wrong with your program, the larger

the portion of credit WE (and not you) earn.

In short, you need to make it easy for us to examine your program.

**2. Your output should match the corresponding tree for each of the test programs.**

**The test programs are in directory tiny\_test\_progs.**

**There are 25 test programs. Each is accompanied by the corresponding AST,**

**generated by our parser.** Please bear with the idiosyncracies of our parser,

and match the tree format \*EXACTLY\* (down to the very last character).

**3. Grading strategy:**

We will run your program on each of the test programs, and we will compare (bit by bit,

using ‘diff’) ) your AST with the correct one. Full credit will be given a perfect match in

every case. Credit will be lost for failing to follow the above instructions.

Readability and clarity of the source code is also important for your grade.

**4. Executable Name:**

The executable file should be named ‘subc’.

You must provide a makefile to build this project. We will unpack, compile and run

your program (on thunder.cise.ufl.edu) as shown below. You \*MUST\* make this

sequence work for us. NO ADDITIONAL SYSTEM TOOLS ARE ALLOWED.

> tar xvf <your\_submission\_file>.tar

>make

> subc –ast tiny\_test\_progrs/tiny\_01 > tree.01

> diff tree.01 tiny\_test\_progs/tiny01.tree

>subc –ast tiny\_test\_progs/tiny\_02 > tree.02

> diff tree.02 tiny\_test\_progs/tiny02.tree

…

Notice that the input is NOT standard input (stdin), but instead a command-line

parameter.

**5. subc (without switches) should produce no output whatsoever for a syntactically**

**correct program, and a suitable error message for an incorrect one.**

Required switches: -ast. This switch prints the abstract syntax tree, and nothing else.

No headers or footers.

**6. It must be possible to run subc from ANY directory, i.e. the executable, once**

**compiled, must be completely self-contained, and not depend on any files or**

**data in the current directory.**

**For example,**

>~user/cop4020/subc my\_test\_progs/testfile

should run correctly.