COMSC 440 Language Translation and Compiler Design

Spring 2020

Project 3: C-Minus parser

Due Date: Friday**, April 24th,** before 12pm.

Here are instructions for your C-Minus parser project. Make sure you read the description of the C-Minus grammar (EBNF notation) on pages 492-493 of the textbook. You are to create a (scanner + parser)-only version of the C-Minus compiler by using the TINY compiler as your model.

This is a group project with two to three persons in each group. Stay with the same group as in Project 2.

Step 1. Backup all your materials from project 2 and copy them to a new folder called project3. Download the *parse.h* and *parse.c* to this folder from the Bridges.

Step 2. Complete the *parse.c* program to construct a syntax tree by implementing the recursive-descent parsing algorithm . Refer to the TINY compiler parser design. Put comments to all the codes you add (listed as ****add code here). Also, add the purpose of the program and your own personal information such as name, email, course, and date to the header comments for parse.c and main.c. Modify your main program to allow the parsing procedure happen. Print out the syntax tree y calling the printTree method in util.c. A syntax tree structure for C- is designed for you and the data type declaration for the tree node is shown in global.h for your reference.

Step3. Create a Makefile file to compile and link all the related files.

Step 4. Use the given gcd.cm file (a sample C-Minus program) to test your parser. The correct output is shown as follows and also in the file output:

```
>>>>>C-MINUS COMPILATION: gcd.cm

Syntax tree:
  fun: gcd: int
    param: u: int
    param: v: int
    Cmpd
      If
        Op: ==
          Id: v
          Const: 0
        Return
          Id: u
        Return
          Call: gcd
            Id: v
            Op: -
```

```
            Id: u
            Op: *
              Op: /
                Id: u
                Id: v
              Id: v
fun: main: void
  Cmpd
    var: x: int
    var: y: int
    Op: =
      Id: x
      Call: input
    Op: =
      Id: y
      Call: input
    Call: output
      Call: gcd
        Id: x
        Id: y
```

Step 5. Compress all eight source code files (i.e. the files **`globals.h`**, **`util.h`**, **`scan.h`**, **`main.c`**, **`util.c`**, **`scan.c, parse.h and parse.c`**), together with the test file gcd.cm and the Makefile file into a file named project3XXX, where XXX are your group initials and upload them to the Bridges under the folder of project3.

Step 6. A report is required for each group to show the implementation details and the group member assignments. A ***syntax tree*** (not the result by calling printTree procedure) for the *gcd.cm* program (follow the example in Fig 3.9 on p138 and use the syntax tree design given in global.h) should be included in your report. A hard copy of your obtained syntax tree and the report are due on April 24th, before 12pm.