Programming Assignment 0 – Due Sept 26 at 11:59PM

This assignment is intended to demonstrate how context free grammars are parsed and the submission process for turning in assignments.

1. Install Software

First, download and install a Java IDE that you like – NetBeans, Eclipse, IntelliJ, or whatever you prefer.

2. GitLab

In this class, we are going to use GitLab[1] to distribute starter code, as well as for you to submit lab and programming assignments. GitLab is one of several popular web-based platforms (e.g. GitHub, BitBucket) that provide *version control*[2] services and software. While it is common to use a cloud-based version, in this class we'll use a WIT-based installation.

We won't talk about version control much in this class, but it's basic function is to manage *changes* to documents (in this case, the source code for your assignments) – it is crucial for most software projects, especially those involving teams, and is a fundamental skill for software development in industry. FYI: there are many software packages used for revision control (e.g. CVS, Subversion, Mercurial) – GitLab, as the name implies, uses git[3], a free and open-source system that is useful and popular for personal and industrial software development.

2.1. Login

Visit the GitLab website and use the forms there to create your account **using your WIT email**. If you already have a GitLab account you are welcome to use it, or to create one specifically for this class. Feel free to supply customize your profile/avatar as you wish, and set privacy/account settings as you see fit.

Important: Use Your WIT E-mail as Primary
In this class we will be automating the process of downloading and grading your programming/lab. This process depends upon you using your WIT e-mail (whatever@wit.edu) as your primary GitLab e-mail address. Thus, on the profile page, make sure your WIT e-mail appears in the Email field.

3. Complete the Assignment

The steps up till now you only need to do once. By contrast, you will follow the remaining steps for each assignment for the rest of the semester. Before you begin, here is the big picture of the process you will complete for each assignment:

[1] https://gitlab.com
[2] https://en.wikipedia.org/wiki/Revision_control
[3] https://git-scm.com

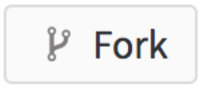
1. Visit the assignment **repository**
2. **Fork** the repository (i.e. make a *private* copy that is owned by your account)
3. Import or download the repository
4. Write your code, saving your work frequently (**commit** and **push**)
5. Confirm your final submission is accessible to the instructor
6. Set appropriate **access** to your forked repository (add the instructor)

3.1. Visit the Assignment Repository

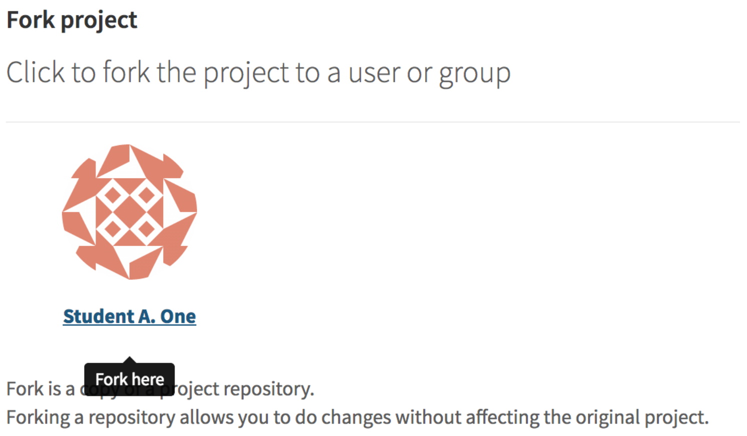
In most version control systems there is the concept of a *repository* (or "repo" for short), which is a collection of one or more files and folders. In this class, each assignment will have its own repository. In GitLab, and comparable services, each repository has its own corresponding website. So to begin, visit the website for this assignment's repository (a link will be listed under "Your Projects" when you visit the GitLab main dashboard – accessible by clicking the Wentworth logo at the top-middle of any page).

3.2. Fork the Assignment Repository

On the repository website, click the "Fork" button – this will create a copy of the repository that your account will own (this will be where you will submit your work).



The resulting page will ask which user/project should own the forked repository – click the box that represents your user.


Fork project

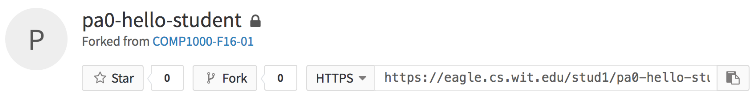
Click to fork the project to a user or group

Student A. One

Fork here

Fork is a copy of a project repository.
Forking a repository allows you to do changes without affecting the original project.

If successful, you will be brought to the website of your forked repository (note that it will include a new button showing the repository from which it was forked).


pa0-hello-student

Forked from COMP1000-F16-01

https://eagle.cs.wit.edu/stud1/pa0-hello-stu

•

3.3. Download/Import files

You now have a private copy of the assignment in your repository on GitLab's servers – it's time to bring this to your computer, so you can write some Java code! Most IDEs have a method for integrating with a git repo. You can either use that, another graphical front end to git, or git itself from the command line.

3.4. Written Code

See section 4 for specifications about this assignment.

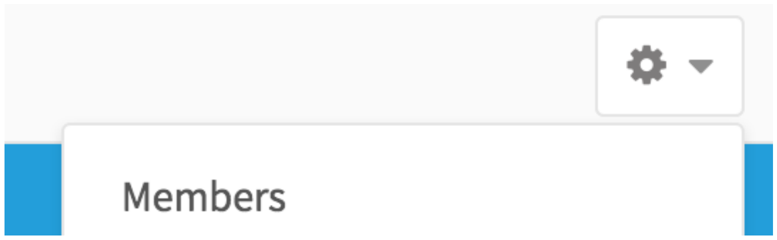
3.5. Confirm Submission

Now complete this assignment, as you will EVERY assignment in this class:

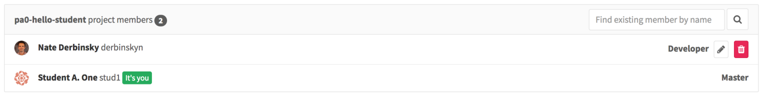
- Save, commit, and push all code changes
- Confirm the latest code is visible via the "Files" section of your repository website
- Add instructor access to your project (See section below)

3.6. Set Repository Access

The lock icon next to the name of the repository means that no one can access your repository unless you provide explicit access. To grant access, click the "Members" link from the "gears" (i.e. settings) menu:



On the resulting page, find your instructor (kreimendahlf) in the **People** field, set **Project Access** to Developer, and click the "Add users to project" button. If this succeeds, you should see your user and the instructor, tagged as a Developer. Submission consists of you adding your instructor to the project.


pa0-hello-student project members 2
Find existing member by name
Nate Derbinsky derbinskyn
Developer
Student A. One stud1 It's you
Master

<u>**Remember**</u>

- If your repository is shared with any additional users, we will assume that you were cheating by letting other students see your work.
- If your instructor does not have Developer access, your work cannot be graded and you will receive a 0.

4. Specifications

The purpose of this assignment is to implement an LR parser that takes a grammar and a string as inputs, and outputs whether the string is valid in the grammar.

4.1. Language

This assignment should be completed in Java. Future assignments will be written in different languages, but this one must be completed in Java, using any IDE that you are comfortable with. Please add sufficient comments to your code so that it's clear to a smart but uninformed reader what your intention is with a code block.

4.2. Input

The program will take two inputs, which will be stored in text files. These two files will be specified as command line arguments with the following usage: `java com.namespace.your.Parser grammar.txt string.txt`. The grammar will always be the first argument, and the string to validate will be the second argument. The two input files will have guaranteed structures. Make sure that you check that the files exist before trying to open them, but assume that they have proper structure once they are open.

The grammar file will be structured with the following guarantees:

- All tokens will be separated with spaces.
- One rule will appear on each line.
- The start rule will be on the first line of the file.
- Each line with either start with a nonterminal token or a '|' character. If the line starts with the '|' character, the left hand side of that rule is the same as on the previous line.
- There will be at least one rule.
- The grammar will be parsable with an LR parser.
- The file will end with a newline.

The string file will be structured with the following guarantees:

- All tokens will be separated with spaces.
- All tokens will match a terminal in the grammar.
- The string will be a single line of text and end with a newline.

4.3. Parser

The parser should implement the shift-reduce parser algorithm, as presented in class. The class name should be Parser. Any supplemental classes or other data structures that you choose to implement or import are up to you.

4.4. Output

The program has two possible outputs (assuming no file reading exceptions or other catastrophes): If the input string is valid in the grammar, output "string is valid!". Otherwise, print out "string is invalid!". Remember, the string is only valid if you parse the entire string and the stack contains only a single start token.

4.5. Testing

Included in the initial repo is a directory of grammars and strings. Your assignments will be tested on these files, as well as a few others. It is probably a good idea to write some of your own test files to verify that your program runs correctly.